5-2005

Energy Efficient Designs for Collaborative Signal and Information Processing inWireless Sensor Networks

Yingyue Xu
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a dissertation written by Yingyue Xu entitled "Energy Efficient Designs for Collaborative Signal and Information Processing inWireless Sensor Networks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Electrical Engineering.

Hairong Qi, Major Professor

We have read this dissertation and recommend its acceptance:

J. Douglas Birdwell, Itamar Elhanany, Lynne E. Parker, Stephen Fulton Smith

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Yingyue Xu entitled "Energy Efficient Designs for Collaborative Signal and Information Processing in Wireless Sensor Networks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Electrical Engineering.

Hairong Qi
Major Professor

We have read this dissertation
and recommend its acceptance:

J. Douglas Birdwell

Itamar Elhanany

Lynne E. Parker

Stephen Fulton Smith

Accepted for the Council:

Anne Mayhew
Vice Chancellor and Dean of
the Graduate Studies

(Original signatures are on the files with official student records.)

Energy Efficient Designs for Collaborative Signal and Information Processing in Wireless Sensor Networks

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Yingyue Xu

May 2005

Acknowledgments

First, I would like to express my deepest gratitude my adviser Dr. Hairong Qi. Her valuable guidance and patience through my studies has allowed me to develop my skills as a researcher. Whenever I met difficulties, whether in my research or personal life, she provided steadfast support and encouragement. Without her guidance, this work would not have been possible. Also I am highly indebted to Dr. Birdwell, Dr. Elhanany, Dr. Parker and Dr. Smith for many valuable discussions, insightful suggestions and serving as my committee members.

Many thanks are to my peers, Yang Liu, Hongtao Du, Xiaoling Wang, Lidan Miao, Sankar Venkatraman, Balasubramanian Lakshminarayanan, Olawoye Oyeyele, Phani Teja Sastry Kuruganti, and Gaurav Baone, for their friendship and support.

Last but not least, I sincerely thank my parents, my sister, and my brother-in-law. Your support, love and enthusiasm through my life make everything possible and meaningful.

Abstract

Collaborative signal and information processing (CSIP) plays an important role in the deployment of wireless sensor networks. Since each sensor has limited computing capability, constrained power usage, and limited sensing range, collaboration among sensor nodes is important in order to compensate for each other's limitation as well as to improve the degree of fault tolerance. In order to support the execution of CSIP algorithms, distributed computing paradigm and clustering protocols, are needed, which are the major concentrations of this dissertation.

In order to facilitate collaboration among sensor nodes, we present a mobile-agent computing paradigm, where instead of each sensor node sending local information to a processing center, as is typical in the client/server-based computing, the processing code is moved to the sensor nodes through mobile agents. We further conduct extensive performance evaluation versus the traditional client/server-based computing. Experimental results show that the mobile agent paradigm performs much better when the number of nodes is large while the client/server paradigm is advantageous when the number of nodes is small. Based on this result, we propose a hybrid computing paradigm that adopts different computing models within different clusters of sensor nodes. Either the client/server or the mobile agent paradigm can be employed within clusters or between clusters according to the different cluster configurations. This new computing paradigm can take full advantages of both client/server and mobile agent computing paradigms. Simulations show that the hybrid computing paradigm performs better than either the client/server or the mobile agent computing.

The mobile agent itinerary has a significant impact on the overall performance of the sensor network. We thus formulate both the static mobile agent planning and the dynamic mobile agent planning as optimization problems. Based on the models, we present three itinerary planning algorithms. We have showed, through simulation, that the predictive dynamic itinerary

performs the best under a wide range of conditions, thus making it particularly suitable for CSIP in wireless sensor networks.

In order to facilitate the deployment of hybrid computing paradigm, we proposed a decentralized reactive clustering (DRC) protocol to cluster the sensor network in an energy-efficient way. The clustering process is only invoked by events occur in the sensor network. Nodes that do not detect the events are put into the sleep state to save energy. In addition, power control technique is used to minimize the transmission power needed. The advantages of DRC protocol are demonstrated through simulations.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Sensors and actuators serve as the interconnection between human being and the physical world. Sensors *detect* the physical nature of the world, such as the light, the temperature, or the sound. Similarly, actuators *affect* the world in some way, such as toggling a switch, making a noise, or exerting a force. Such a close relationship with the physical world reflects a dramatic contrast to traditional computing, which deals exclusively with the information generated by humans, such as e-mail, digital music, or bank balances. Recent advances in Micro-Electro-Mechanical Systems (MEMS) and wireless communications have fomented the development of low-cost, low-power, untethered, tiny sensor nodes, equipped with significant processing, memory, and wireless communication capabilities. These sensor nodes are capable of monitoring a wide variety of ambient conditions, such as temperature, pressure, mechanical stress level on attached objects, etc. Such sensor devices can be densely and randomly deployed to form a new kind of network – the wireless sensor networks (WSNs).

Researchers on WSNs have drawn promising pictures about the future of wireless sensor networks. In an interesting article [135], M. Weiser describes the world in the future, where "ubiquitous computers will come in different sizes, each suited to a particular task","hundreds of computers in every room, all capable of sensing people near them and linked by high-speed

networks". Furthermore, "Ubiquitous computing help overcome the problem of information overload." "Machines that fit the human environment, instead of forcing humans to enter theirs, will make using a computer as refreshing as taking a walk in the woods." In an imaginary earthquake hit in Southern California in 2053 [43], J. Elson and D. Estrin describe how Southern California return to normal under the help of "pervasive sensors that had been woven into both technological fabric and the natural environment". "Sensors throughout the city's water system went onto high alert, ready to divert contaminants to a safe disposal if any toxins were detected." "Small fire was detected and a map to the area popped up on a screen." Even though such technology seems fanciful today, there is no doubt that wireless sensor networks will become ubiquitous and change people's life dramatically.

This dissertation presents the details of research at the University of Tennessee in wireless sensor networks and their applications in collaborative information processing. The remainder of this chapter first gives an overview of sensor nodes as well as the wireless sensor network, including the architecture of the sensor node, the various existing sensor node designs, the challenges posed on the research of the wireless sensor networks, and the applications of WSNs. It then focuses the discussion on the issue of energy efficiency in WSNs and provides an overview of existing protocols designed in different layers for energy efficiency purpose. It finally addresses the application layer and introduces some background knowledge of collaborative signal and information processing (CSIP) and computing paradigms that facilitate CSIP in wireless sensor networks, including the client/server paradigm and the mobile agent paradigm.

1.1 Sensor Nodes

A sensor node is the basic element in wireless sensor networks. It is a battery-driven device, consisting of a sensing unit, a processing unit, a transceiver and a power unit. Some type of nodes may also have a power generator, a location-finding system and a mobilizer. See Fig. 1.1 for the architecture of a sensor node.

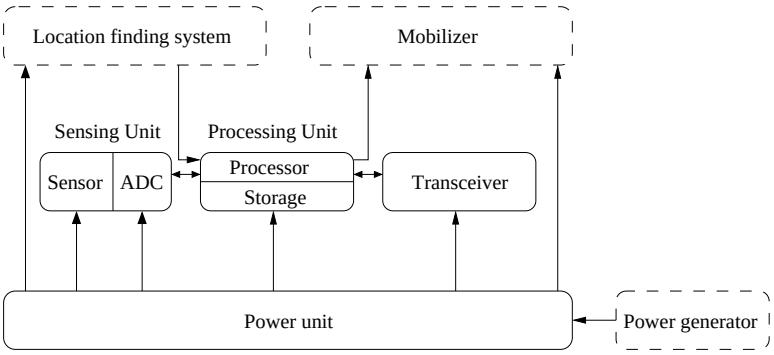


Figure 1.1: Sensor node architecture (redrawn from [19]).

The *sensing unit* consists of various sensors and analog-to-digital converters (ADC). The *sensors* are transducers converting the physical phenomena into electrical signals. These sensors may include, for example, a microphone, which detects acoustic signals and produces an electrical image of the sound; a geophone, which measures the earth's vibrations at a single point; an accelerometer, which produces a voltage proportional to the acceleration of the ground vibrations; or a passive infra-red (PIR) detector, which picks up the infrared (IR) radiations emitted by moving objects that have detectably different temperatures from the surrounding ambient temperature. All these signals from the physical world decay as the distance to the source increases.

The *processing unit* is responsible for processing the digital signal and executing various algorithms. For example, Sensoria nodes [14] use the Hitachi SH4 processor, a powerful 167 MHz processor that can handle complex algorithms. On the contrary, the Berkeley's Motes [2] use a lightweight Atmel AVR8535 at 4 MHz, which has high energy efficiency but much less computing capability. In general, lightweight nodes are only suitable for simple processing, such as the average, minimum, and maximum values from different sensor readings [85], due to their limited computing ability. On the other hand, powerful nodes can perform processing tasks that require extensive computations, such as beam-forming, target classification, and tracking.

The *transceiver* is responsible of communication with other nodes. The use of radio-frequency (RF) communication is preferred in sensor networks because the packet transmission is small, and the bandwidth usage is efficient due to the short range of communication distances [18]. The RF communication involves modulation, bandpass, filtering, and demodulation circuitry. Underwater sensor networks [97] use acoustic frequencies because of the special propagation environment.

The *power unit* provides the energy source for the sensor node. It is usually in the form of batteries. Once used up, the batteries are usually difficult to be replaced. Sometimes it may also consist of a power generator unit, such as a solar cell to supply the energy of the node. The power unit determines the lifetime of the node and is the most important unit.

The *location-finding system* provides the location information of the node, which is vital to some algorithms. The Sensoria node has a GPS system to find the current position of the node.

The *mobilizer* is useful when the sensor node needs to move to other place in some applications. It is, of course, an optional component.

Currently, there are considerable research programs conducted on sensor node hardware development. Some of their products are:

- **UC Berkeley Motes**: Motes (including Mica, Mica2, Dot and Spec) are designed by the University of California, Berkeley and are popularly called "Smart Dust" [65]. The Dot mote is shown in Fig. 1.2 (left). The newest Spec mote is shown in Fig. 1.2 (right). The design goal of the Smart Dust is to build a self-contained, millimeter-scale sensing and communication platform for a massively distributed sensor network. The device will be around the size of a grain of sand. It is a tiny node containing an MCU (ATMEL 90LS8535) [2] and runs at 4 MHz and 3.0 V. It has an 8KB flash memory and a 512-byte SRAM memory. The radio is an asynchronous input/output device with hard real-time constraints. It consists of an RF Monolithics 916.50 MHz transceiver, antenna and collection of physical-layer components to configure the physical layer characteristics

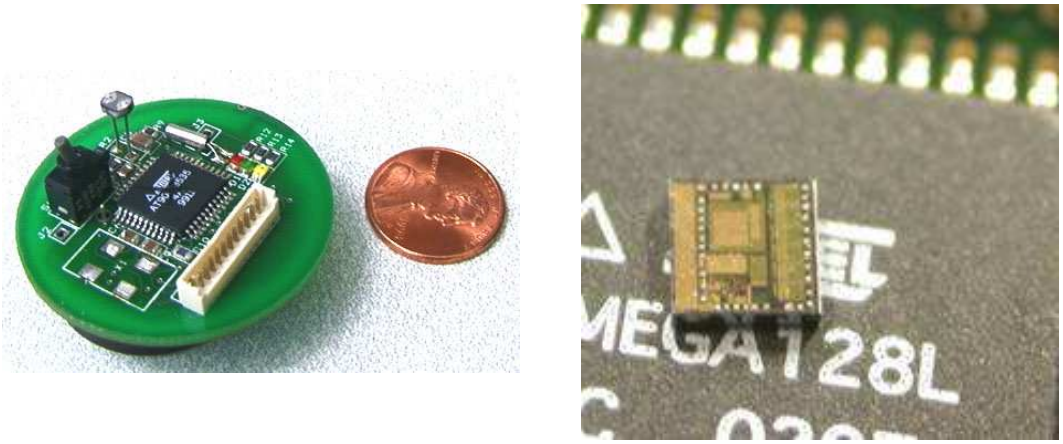


Figure 1.2: UCBerkeley Mote: Dot and Spec [11].

such as signal strength and sensitivity. It comes with a temperature sensor with an option to mount custom-selected sensors on the sensor board. The nodes run the TinyOS [13] operating system, which supports two-level scheduling and allows for high concurrency to be handled in a very small amount of space [65].

- **Sensoria WINS NG nodes**: The WINS NG node is a powerful sensor node platform with several interfaces to externally connect sensors, wireless extension cards, and other serial port devices. Fig. 1.3 shows a Sensoria sensor node. The node uses the Hitachi SH-4 processor running at 167 MHz. The SH-4 is a 32-bit RISC with a 128-bit vector floating point unit (FPU) and super-scalar implementation providing higher speeds at low clock rates [10]. The sensor node supports four sensing channels, including, for example, acoustic, seismic and PIR sensors. It also hosts a GPS module for geo-location information of the nodes. It has dual RF modems, both of which in the 2.401 - 2.495 GHz ISM band using frequency-hopping spread spectrum (FHSS). The node runs a Linux

Figure 1.3: Sensoria WINS NG 2.0 node [116].

kernel as the operating system. Sensoria provides the APIs [14] required for RF modem control and data acquisition.

- **MIT μAMPS-we and II nodes**: MIT's μAMPS project is a research effort focused on energy efficiency design at all levels of the sensor networks system. The μAMPS-we is their first step towards building a wireless sensor network. It is implemented with commercial off-the-shelf (COTS) components. Fig. 1.4 shows the outlook of the node. μAMPS-II is the ongoing project to build energy-scalable micro-power DSP at 10 MIPS. The node will have two dedicated ASICs, one for digital processing and one for the analog/RF part of the radio.

- **PC-104 based nodes**: The term PC-104 is derived from the connector used to stack different boards having 104 pins. PC-104 is an industry standard of PC-compatible modules that can be stacked together to form a custom-designed embedded system [3]. Since these systems are made with hardware, compatible with PC systems, it is easy to configure them along with the PCs. The PC-104 sensor nodes are custom built with chosen processor, memory configuration and hard disk. Fig. 1.5 shows a PC-104 based sensor node. The SCADDS testbed of USC/ISI consists of 30 nodes built using PC-104 based products [4].

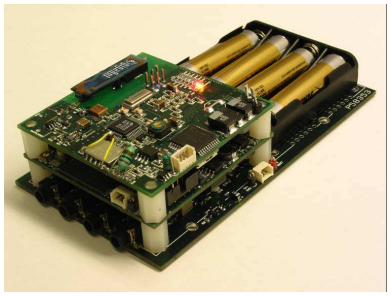


Figure 1.4: μAMPS-we node [6].

Figure 1.5: PC-104 based sensor node.

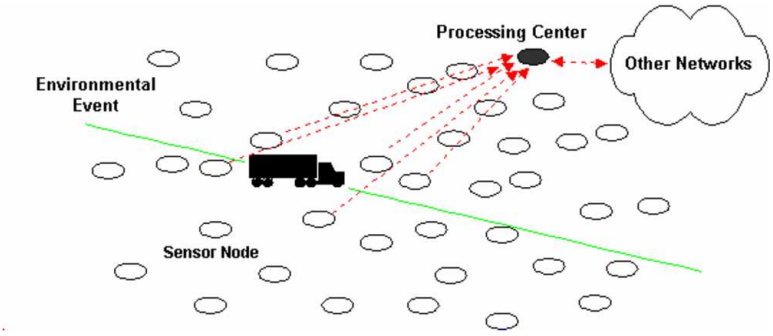


Figure 1.6: A wireless sensor network.

1.2 Wireless Sensor Networks: Promises and Challenges

A wireless sensor network is a special kind of network that consists of sensor nodes. A typical wireless sensor network looks like Fig. 1.6, where sensor nodes are scattered in the area of interest. When certain events occur, the alerted sensors will collect data and send it to a processing center. The processing center processes the data and generates the results, which can be accessed by users through other networks, such as the Internet.

There are a wide spectrum of applications for wireless sensor networks, ranging from military to civilian and medical.

For military applications, WSNs are useful in battlefield surveillance, target classification and tracking, monitoring and reconnaissance, and the like. They can also be effectively used to sense the chemical level in the environment in the event of biological or chemical attacks. WSNs can be an integral part of military *command, control, communications, computers, intelligence, surveillance, reconnaissance and tracking* (C4ISRT) systems. The rapid deployment, self organization and fault-tolerance characteristics of sensor networks make them a very promising surveillance technique for military C4ISRT. Since sensor networks are based on the dense deployment of disposable and low-cost sensor nodes, destruction of some nodes by an enemy does not affect a military operation as much as the destruction of a traditional sensor. The Sensor

Information Technology (SensIT) project sponsored by Defense Advanced Research Projects Agency (DARPA) [9] is one of the first-initiative programs in wireless sensor network technologies that seeks applications in the future battlefield.

Environmental applications may include tracking the movements of animals; monitoring marine, soil, and atmospheric conditions; forest fire detection; macroinstruments for large-scale Earth monitoring and planetary exploration; flood detection, and so forth [16, 70, 74, 122, 134]. The Habitat Monitoring on Great Duck Island by the Intel Research Laboratory at Berkeley, College of the Atlantic, and the University of California at Berkeley [86] monitors the microclimates in and around nesting burrows used by the Leach's Storm Petrel in Great Duck Island, Maine. The goal is to develop a habitat monitoring kit that enables researchers worldwide to engage in the non-intrusive and non-disruptive monitoring of sensitive wildlife and habitats.

Wireless sensor networks may also be used in medical applications. Some of the potential applications include remote patient monitoring, drug administration and in-hospital environment control. For example, the University of Notre Dame, Brunel University, and the University of Miami are conducting a Mobile Patient project [27] that uses sensor networks for patient monitoring and care.

Residential and commercial applications for wireless sensor networks include home automation and smart environment. Some of them include building virtual keyboards; constructing smart office spaces; and environmental control in office buildings [44, 74, 96, 102]. The sensor nodes can be embedded into furniture or clothes, such as in the Smart Kindergarten project by UCLA [124]. This project targets the early childhood education environment as a testbed in order to provide parents and teachers with the abilities to comprehensively investigate students' learning processes.

Although promising, the unique characteristics of wireless sensor networks also pose unique challenges to traditional network design. In the following, we summarize some important issues brought up by wireless sensor networks [18, 19, 60].

Energy [50, 103]. Sensor devices are battery operated. Once deployed, it is usually impossible to replace the batteries. Therefore, how to save energy, and thus prolong the lifetime of the individual sensor node as well as the whole sensor network, is a major challenge in wireless sensor networks.

Scalability [45]. A huge amount of sensor nodes could be deployed to form a wireless sensor network. For such a large-scale network, the many protocols proposed for ad hoc networks may not be suitable. Innovative scalable algorithms and protocols need to be developed.

Reliability [125]. Sensor nodes are often deployed in harsh, dangerous, or inaccessible environments, such as battlefields or underwater. Such environments make sensor nodes prone to failures. They can also inoperative because of battery depletion. Furthermore, the wireless transmission in sensor networks has a high bit-error rate (BER) and low bandwidth. In such a dynamic network environment, system fault-tolerance and reliability is another challenge.

Real-time operation [84]. Although real-time performance is not the first priority, in some applications, emergent information has to be provided in a timely manner. Thus, how to reduce latency in the presence of limited energy is also worth studying.

1.3 Energy Efficiency in Wireless Sensor Networks

Energy optimization in the case of wireless sensor networks is much more complex than conventional low-power design techniques [33], because it involves not only reducing the energy consumption of a single sensor node, but also maximizing the lifetime of an entire network. The energy efficient designs should be incorporated into every stage of the wireless sensor network design and operation in order to prolong the overall network lifetime.

1.3.1 Source of Power Consumption

In order to design energy-efficient systems, we need to first investigate the sources of power and dissipation characteristics of a wireless sensor node.

First, we consider the microcontroller unit. The microcontroller unit (MCU) is responsible for control of the sensors and the execution of communication protocols and signal processing algorithms on the gathered sensor data, thus giving intelligence to the node. The commonly used MCUs are Intel's StrongArm microcontroller and Atmel's AVR microcontroller. Some research has been conducted to estimate the power consumption of these embedded processors [120, 129]. The choice of MCU is related to the application scenario. The StrongARM microprocessor consumes around 400 mW of power while executing instructions, whereas the ATmega103L AVR microcontroller consumes only around 16.5 mW, but provides much lower performance. MCUs usually support various operating modes, including Active, Idle, and Sleep modes, for power management purposes. Each mode is characterized by a different amount of power consumption. For instance, the StrongARM consumes 50 mW of power in the Idle mode, and only 0.16 mW in the sleep mode. However, transitioning between modes involves a power and latency overhead. Thus, the power consumption levels of various modes, the transition costs, and the amount of time spent in each mode all have a significant effect on the total energy consumption of the microcontroller.

Second, we consider the transceiver unit. Several factors affect the power consumption characteristics of a transceiver, including the type of modulation scheme used, data rate, RF transmission power (determined by the transmission range), and the operational duty cycle. Radios can also operate in four distinct modes: Transmit, Receive, Idle and Sleep. Xu et al. [139] shows that the Idle mode consumes significantly higher power than the Sleep mode, almost equal to the power consumed in the Receive mode. Thus, we should make the radio stay in the Sleep state instead of the Idle mode whenever possible. The change from one mode to another mode causes a comparatively large amount of power dissipation. For example, when the radio switches from sleep mode to transmit mode in order to send a packet, a significant amount of power is consumed for starting up the transmitter [131]. When designing protocols, we should consider all these facts and avoid unnecessary changes of modes.

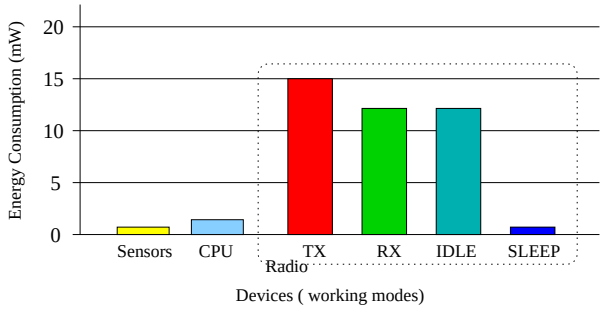


Figure 1.7: Comparison of energy consumption of different devices of the general sensor node [46].

Third, we consider the sensing unit. Sensors translate physical phenomena to electrical signals and can be classified as either analog or digital devices, depending on the type of outputs they produce. There are several sources of power consumption in a sensor, including: signal sampling and conversion of physical signals to electrical ones; signal conditioning; and analog-to-digital conversion. Given the wide diversity of sensor types, there is no typical power consumption number. In general, the energy consumed by the sensors is small. However, active sensors such as sonar rangers, and array sensors such as imagers can be large consumers of power.

Fig. 1.7 shows the energy consumptions of different components on a sensor node. We can conclude that the largest part of the energy consumption goes to the Radio. More specifically, Table 1.1 gives the power consumption of Rockwell's WINS node, which is equipped with a StrongARM SA-1100 processor, a radio module from Conexant Systems, and acoustic and seismic sensors. We can see that the node power consumption is strongly dependent on the operating modes of the components. For example, as Table 1.1 shows, the WINS node consumes only around one-sixth the power when the MCU is in the Sleep mode, compared to the Active mode. We also notice that the transmission power can be adjusted to different levels of energy consumption.

Table 1.1: Power analysis of Rockwell's WINS nodes [47].

MCU mode	Sensor Mode	Radio Mode	Power (mW)
Active	On	T_x(Power:36.3 mW)	1080.5
		T_x(Power:19.1 mW)	986.0
		T_x(Power:13.8 mW)	943.6
		T_x(Power:3.47 mW)	815.5
		T_x(Power:2.51 mW)	807.5
		T_x(Power:0.96 mW)	787.5
		T_x(Power:0.30 mW)	773.9
		T_x(Power:0.12 mW)	771.1
Active	On	R_x	751.6
Active	On	Idle	737.5
Active	On	Sleep	416.3
Active	On	Removed	383.3
Sleep	On	Removed	64.0
Active	Removed	Removed	360.0

Table 1.2: Comparison of different protocols.

Protocol	Layer	Energy-saving method
LEACH [62]	Application	Clustering, cluster head rotation
STEM [114]	Application	Topology management, two channels
Directed Diffusion [69]	Routing	Data centric, receiver initiate
SPIN [60]	Routing	Data centric, meta-data, sender initiate
SAR [147]	Routing	Minimum QoS metrics
SMAC [146]	MAC	Periodic listen and sleep, collision and overhearing avoidance
B-MAC [32]	MAC	Upcalls
PAMAS [119]	MAC	Two channels
EAR [130]	MAC	Registry
DPM [29]	Physical	Idle component shutdown
DVS [89]	Physical	Adapting supply voltage
TinyOS [13]	Operation system	Event-based approach

1.3.2 Power-Save Protocols

Power-save protocols seek to maximize energy saving, while minimizing their impacts on throughput and latency. According to different functions, we can divide the wireless sensor network protocols into application layer, network layer, MAC layer, physical layer and operating system (OS) layer. The sensor network lifetime can be significantly enhanced if all these layers are designed energy-efficiently. Here, we briefly review the state-of-the-art energy efficient designs in wireless sensor networks in Table 1.2.

Application Layer

Low-Energy Adaptive Clustering Hierarchy (LEACH): LEACH [60] is a cluster-based protocol that divides the sensor network into clusters, thus allowing nodes that are close to each other to share data. The nodes in the cluster send their data to a local *cluster-head*, which is responsible for receiving all the data from nodes within the cluster and aggregating this data

into a smaller set of information that describes the events the nodes are sensing. The operation of LEACH is divided into two phases: the setup phase and the steady phase. During the setup phase, each node generates a random number and compares it to a threshold to decide whether it becomes a cluster head or not. Once the cluster heads are determined, they will announce their decisions to the whole network. The sensors receiving the announcements will determine which cluster to join based on the signal strengths of the announcements. LEACH uses the following techniques to achieve energy and latency efficiency: (i) use of a cluster-formation algorithm which allows each node to make autonomous decisions that result in good clusters being formed; (ii) localized control for data transfer; (iii) use of local control to set up a TDMA schedule and implementing low-energy media access; and (iv) application-specific data processing, such as data aggregation or compression to reduce the amount of data need to be transmitted.

Sparse Topology and Energy Management (STEM): STEM [114] is a new topology management scheme. It trades off energy consumption in the monitoring state, a state to monitor its environment, versus latency of switching back to the transfer state, which is the state to forward data. It emulates a paging channel by having a separate radio operating at a lower duty cycle. Upon receiving a wakeup message, it turns on the primary radio, which takes care of regular data transmissions. STEM is specifically geared towards those scenarios where the network spends most of its time waiting for events to happen without forwarding traffic.

Sensor Query and Tasking Language (SQTL): SQTL [117] is a language providing a large set of services. It supports three types of events, which are defined by keywords *receive*, *every*, and *expire*. *Receive* defines events generated by a sensor node when the sensor node receives a message; *every* defines events occurring periodically due to a timer time out; and *expire* defines the events when a timer is expired.

Ghaisi et al. [54] study the theoretical aspects of the clustering problem in wireless sensor networks with application to energy optimization. The authors illustrate an optimal algorithm

for clustering the sensor nodes such that each cluster is balanced and the total distance between sensor nodes and the cluster heads is minimized. It is helpful in reducing the communication overhead and hence the energy dissipation. The authors solve this optimization problem by modeling it as a minimum-cost flow problem.

Network Layer

Youssef et al. [149] introduce a new energy-aware routing protocol that tries to minimize the energy consumption and at the same time maintain good end-to-end delay and throughput performance. The protocol is based on a constrained shortest-path algorithm. The paper shows that for large values of the constraint, the algorithm gives performance similar to the direct routing algorithm and for moderate values, the performance is acceptable under all performance metrics and presents a balance between the minimum transmission energy routing algorithm and the direct routing algorithm.

Directed Diffusion: Directed diffusion [45] is a novel routing protocol suitable for event-driven applications like wireless sensor networks. It is "data-centric", meaning routing is based on data contained in the sensor nodes rather than traditional IP address. The data is identified by its attributes in directed diffusion. Sinks or nodes that request data send out *interests*. If the attributes of the data match these interests, a *gradient* is setup and data will be pulled to the sinks. It provides a Geographic and Energy Aware Routing Protocol (GEAR) which is energy efficient by using geographic regions and avoiding blind flooding.

Sensor Protocols for Information via Negotiation (SPIN): SPIN [62] is a routing protocol designed for wireless sensor networks. SPIN names its data using high-level data descriptors, called the *meta-data*, which have a one-to-one mapping relation with the raw data. In the SPIN protocol, the node which has new data *advertises* the data to the neighboring nodes in the network using meta-data. When the neighboring node wants the data, it sends a *request* to the initiator node, which will *send* data to the sinks. Each node has its own resource manager

to keep track of the usage of energy resource and make decisions such as whether it should participate in transmission accordingly.

Sequential Assignment Routing (SAR): SAR [123] takes into consideration the energy resources and QoS on each path, and the priority level on each packet for making routing decisions. A multi-path approach is used to avoid the overhead of route recomputation on failure and localized path restoration schemes. To create multiple paths from each node to the sink, multiple trees, each rooted from a one-hop to the sink, are built. Each tree is grown outward from the sink by successively branching and going for higher hop paths while avoiding nodes with lower QoS and energy reserves. At the end of this process, each node will belong to multiple paths and each sensor can control which one-hop neighbor of the sink will be used for relaying the message. An additive QoS metric and a measure of the energy resources are associated with each path. SAR calculates a weighted QoS metric as the product of the additive QoS metric and a weight coefficient associated with the priority level of the packet. A periodic recomputation of paths is triggered by the sink to account for any changes in topology. Failure recovery is done through a handshaking procedure between neighbors and local path restoration is used.

Minimum Cost Forwarding Algorithm: Ye et al. [145] exploits the fact that data flows in sensor networks are in a single direction and are always to the fixed base station. Their method resembles the natural gravity field that drives waterfalls from top of the mountain to the ground. Each node maintains the least cost estimate from itself to the base station. Each message to be forwarded is broadcasted by the node. On receiving a message, the node checks whether it is on the least cost path between the source and the base station. If so, it would forward the message.

MAC Layer

Self-Organizing Medium Access Control for Sensor Network (SMAC): SMAC [146] is a new MAC protocol explicitly designed for wireless sensor networks. While the primary goal

in the design of SMAC is reducing energy consumption, it also achieves good scalability and collision avoidance by utilizing a combined scheduling and contention scheme. It consists of three major components: periodic listen and sleep, collision and overhearing avoidance, and message passing. Each node goes to sleep periodically, and then wakes up and listens to see if any other node wants to talk to it. Before this, it needs to choose a schedule and exchange it with its neighbors. It performs neighbor discovery in the bootup period. As soon as a new link is discovered, the first time slot during which both nodes are free is assigned a channel and is added permanently to their schedule. As time goes on each node grows its neighbor list by attaching new nodes and eventually all the nodes are connected to each other. The ability to have nonsynchronous scheduled communication enables nodes to form links on the fly. After a link is formed, a node knows when to turn on its transceiver ahead of the time for communication. It adopts a contention-based scheme for collision avoidance. SMAC fragments the long message into many small fragments, and transmits them in bursts to achieve high energy and latency efficiency. It has the ability to make trade-offs between energy and latency according to traffic conditions.

B-MAC: B-MAC [95] is a new CSMA-CA scheme proposed by the University of California, Berkeley. It uses upcalls to notify applications of the underlying channel utilization to enable an application to change its sampling phase, sampling interval, or aggregation methods. It also presents an effective carrier sense algorithm for collision avoidance (CA). The B-MAC protocol is implemented on the UC Berkeley Mica2 platform. Their experiments show that B-MAC outperforms the default 802.11-style TinyOS MAC layer by over 100% in terms of packet throughput and achieves 85% channel utilization. It also shows the need for application coordination in wireless sensor network media access layers.

PAMAS: The PAMAS [119] protocol uses an RTS/CTS-style mechanism with a separate control signaling channel. A node that is waiting to initiate a transmission or is in the process of receiving a transmission causes other nodes to defer their transmissions by generating a busy

tone on the control channel. When a node wakes up, it transmits a sequence of probe messages and awaits a response on the control channel to get the information of the channel. The node turns itself off if a neighbor is transmitting and it has no packets to transmit or if it has a packet to transmit, but a neighbor is receiving. PAMAS is most effective in networks with high density and traffic load.

Eavesdrop-And-Register (EAR) Algorithm: EAR [123] has been designed for communication between mobile nodes and stationary nodes on the ground. The mobile nodes are responsible of connection setup to conserve energy. The mobile node keeps a registry of all the sensors in its neighborhood and makes handoff decisions whenever the SNR drops below a threshold value. The EAR uses the invitation messages broadcast during the bootup period as a trigger. The mobile node eavesdrops on all kinds of messages and forms a registry of all the stationary nodes within hearing range.

IEEE 802.15.4 Standard: The IEEE 802.15.4 [67] is a MAC and Physical layer standard for a low data-rate system, providing a multi-month to multi-year battery life solution with very low complexity. It operates at low data rates of 250 kbps (2.4 GHz), 40 kbps, and 20 kbps (868/915 MHz) in order to enhance the lifetime. It has an extremely low duty-cycle ($<$ 10 ppm) capability. It also supports a "Battery Life Extension" (BLE) mode, in which the CSMA-CA backoff exponent is limited to range 0-2 time slots and greatly reduces receiver duty cycle in low traffic applications.

Physical Layer

In addition to using low-power hardware components during sensor node design, operating the various system resources through the use of dynamic power management (DPM) [29] can also save energy consumption. It is based on idle component shutdown, in which the sensor node, or part of it, is shut down or sent into one of several low-power states if no interesting events occur. Besides DPM, additional energy savings are possible in the active state through the use of

dynamic voltage scaling (DVS) [93]. It is to dynamically adapt the processor's supply voltage and operating frequency to just meet the instantaneous processing requirement, thus trading off unutilized performance for energy saving. Several processors, such as Intel's StrongARM and Transmeta's Crusoe, support such scaling of voltage and frequency.

Shih et al. [118] emphasize the importance of correct and accurate modeling of the underlying hardware on energy efficiency and shows the impact of the hardware on the design of the link, MAC, and physical layers of the protocol stack. Data-link and media-access protocols should adapt parameters of the underlying physical layer in order to minimize energy. The protocols designed without knowledge of the hardware can be energy inefficient.

Warneke et al. [134] discuss transmission media for the *Smart Dust* mote [74]. Two transmission schemes, passive transmission using a *corner-cube retroreflector* and active communication using a laser diode and steerable mirrors are examined. In the former, the mote does not require an on-board light source. A configuration of three mirrors is used to communicate binary information. The latter, however, uses an on-board laser diode to send a tightly collimated light beam toward the intended receiver.

Two modulation schemes, binary and M-ary modulations, are compared in [118]. The authors point out that while an M-ary scheme can reduce the transmission on-time by sending multiple bits per symbol, it causes complex circuitry and increased power consumption. Moreover, the authors formulate these trade-off parameters and conclude that under startup-power dominant conditions, the binary modulation scheme performs better in terms of energy efficiency. So, M-ary modulation outperforms binary scheme only for low startup power systems.

Operating System (OS) and Software Layer

The energy consumption can be greatly reduced through energy-efficient software and operating system design. The operating system has direct control over the underlying hardware resources, so its performance can largely determine the performance of the whole network. Moreover, the

OS is ideally poised to implement DPM and DVS based power management policies, since it has global knowledge of the performance and fidelity requirements of all applications, and can directly control the underlying hardware. The core of the OS is the task scheduler, which is responsible for scheduling a given set of tasks to run on the system.

TinyOS: TinyOS [66] is an operating system specifically designed for network embedded systems. TinyOS has a programming model tailored for event-driven applications as well as a very small footprint with only 400 bytes of code and data memory requirement. It is written in nesC, an extension of C for networked embedded systems, such as Berkeley's Mote sensor node. TinyOS uses the event-based approach to create a system that uses CPU resources efficiently. A complete system configuration consists of a tiny scheduler and a graph of components. An application connects components using a wiring specification. There are two kinds of concurrency in TinyOS: tasks and events. Tasks are a deferred computation mechanism, which run to completion and do not preempt each other. In contrast, events may preempt the execution of a task or another event.

Lee et al. [78] propose an accurate software energy consumption model that combines empirical measurement with a statistical analysis technique. By identifying the factors affecting the energy consumption of software, the model also provides insight information that can be used in program optimization for high energy efficiency.

Raghunathan et al. [104] propose an energy-aware real-time scheduling algorithm, which exploits two aspects about the operating scenario of wireless systems: providing an adaptive power *vs.* fidelity tradeoff.

By transforming application software to be energy scalable, the energy-fidelity can be exploited further. The most significant computations should be performed first, so that terminating the algorithm due to energy constraints does not impact the results severely. Several transforms to enhance node energy efficiency are discussed in [121].

1.4 Collaborative Signal and Information Processing in Wireless Sensor Networks

Collaborative signal and information processing (CSIP) in sensor networks has become an important research field. One of the unique features of wireless sensor network applications is the necessity of collaboration. Each sensor node normally has limited processing limitation, constrained power usage, and limited sensing range. Therefore, collaboration among sensor nodes is important in order to compensate for each other's capability as well as improve the degree of fault tolerance. So, the main concern of CSIP is to develop situational awareness using low-level sensor processing and local exchange of data to reach consensus in the neighborhood about the occurring events. The characteristics of wireless sensor networks bring up some important issues for CSIP in wireless sensor networks, which are summarized as follows:

First, dense deployment of sensor nodes [45, 75]. Since thousands of sensor nodes are usually densely deployed in the field, it is possible for the wireless sensor network to provide dense spatial sampling in multi-modality of phenomena of interest. Therefore, the challenge would be to combine the distributed data, first at each node and then with collaboration among the relevant devices in the network to produce meaningful global results. One of the biggest concerns in this process is the design of scalable CSIP algorithms to combat the large number of nodes.

Second, the asynchronous property [75]. The distributed processing in a wireless sensor network typically is asynchronous, for example, in a sequential fusion center, the data from other sensor nodes may arrive out of order. This makes it necessary to design relevant signal and information processing and fusion algorithms in order to deal with the asynchronous executions.

Third, energy efficiency [50, 103]. Sensor devices are battery operated. Once deployed, it is usually very difficult or impossible to replace the battery. Therefore, how to save energy and

how to prolong the lifetime of individual sensor nodes, as well as the whole wireless sensor network, is a major challenge in wireless sensor network research. The network must optimize the trade-off between fault tolerance and energy efficiency in signal processing, data fusion, querying, and routing tasks in order to meet the energy constraints and at the same time to achieve reliable performance. One measurement of energy efficiency is the *lifetime of the wireless sensor network*, which is the time from node deployment to the time when the first node is out of operation due to energy depletion.

Fourth, the reliability issue [125]. Sensor nodes are often deployed in harsh, disastrous or inaccessible environments. Such environments make sensor nodes prone to failures. In addition, the wireless transmission in wireless sensor networks has high bit-error rate (BER) and low bandwidth. Such dynamic network environments present another challenge to achieve fault-tolerance and reliability.

Fifth, the requirement for progressive accuracy [75]. As mentioned before, the sensor nodes are normally battery powered. The limited energy resource makes it important to develop power-aware signal processing and communication methods to provide progressive accuracy, such that the collaboration process could be terminated anytime (upon achieving the desired accuracy) to conserve energy.

In order to combat the challenges posed on CSIP in wireless sensor networks, more and more researchers have been working on various issues to use CSIP technique for intelligence, surveillance and monitoring applications.

Target Detection: Target detection deals with the problem of detecting the presence of targets in the area of interest. The challenging problem in target detection is *source number estimation*, which is to estimate the number of sources [107]. Several techniques have been proposed to tackle this problem. In [107], the authors show that the principled approaches are superior than heuristic methods. Some examples of principled estimation approaches include the Markov chain Monte Carlo (MCMC) [106], Bayesian estimation method [108], and variational learning

approximation [21]. Wang [133] developed a cluster-based distributed estimation framework. The local estimation is generated within each cluster using a progressive detection approach and *a posteriori* probability fusion algorithm is performed to combine the local results based on Bayes theorem. In contrast to the centralized scheme, the progressive approach sequentially estimates the number of targets based on only local observation. She conducted several experiments and showed that the progressive approach can reduce the amount of data transmission to 8.4% (10 sensors) and 12.14% (15 sensors) compared to the classic approach.

Target Localization: The term localization refers to the collection of techniques that measure the spatial relationships in wireless sensor networks. The target localization technique can be divided into two categories: in active localization, the sensor nodes emit signals into the environment; in passive localization, the sensor nodes passively monitor existing signals in a particular channel [113]. Yao et al. [144] proposed a blind beamforming technique for blind source localization, which is an operation without any *a priori* knowledge of the type of signal emitted. They applied several sub-optimal space-time processing algorithms, such as the closed-form least squares source localization, MUSIC (multiple signal classification) algorithm, and the maximum-likelihood parametric method. They claimed that by using the maximum-likelihood algorithm, they can effectively locate multiple targets with random sensor deployment. Zou et al. [153] proposed an energy aware target localization strategy based on an *a posteriori* algorithm with a two-step communication protocol. The cluster head, with very limited data from sensor nodes, executes a localization procedure to determine the subset of sensors to be queried. As shown in the paper, the approach reduces energy consumption, decreases the latency for target localization, and filters false alarms. Wang et al. from UCLA proposed an entropy-based sensor selection heuristic for localization [132]. The heuristic greedily selects an informative sensor at each step and is proved to be more effective when the optimal candidate sensor is more informative.

Collaborative tracking, classification, and data fusion: Zhao et al. developed an information driven approach to sensor collaboration for target tracking, routing, and sensor querying in wireless sensor networks [75, 82, 83, 152]. The key idea is to introduce an information utility measure to select which sensors take part in the tracking, querying, or routing process. This allows us to maximize information gain while minimizing detection latency and bandwidth consumption. They further develop an information-driven sensor querying (IDSQ) framework that select a sensor which is most likely to provide greatest improvement to the estimation of target state. Guibas proposes a way of collaboration from an interesting angle, which is to sense the environment through relation [57] instead of the extensive and detailed data. It uses high-level description of the task to command selective sensor nodes to sense and communicate. The selection process results in minimizing the computational, communication, and sensing resources. The relation-based approach provides us a new angle to solve collaborative information processing problems. If the information-driven approach can be regarded as using a bottom-up approach, then relation-based approach uses a top-down solution. Researchers from the University of Wisconsin-Madison [39, 79] developed a collaborative signal processing framework for tracking multiple targets in a distributed sensor network. The key components of the framework include event detection, estimation and prediction of target location, and target classification. One of the key characteristics in this framework is that routing of information in a sensor network is geographic centric rather than node centric. Brooks and his colleagues present a distributed entity-tracking framework, which embeds the tracking logic in a self-organized distributed network. Tracking is performed by solving the sub-problems of detection, fusion, association, track formation, and track extrapolation [31]. A unique feature of the system is that it contains a mobile code infrastructure for flexible tasking. Mobile code is a software that is transmitted across the sensor network from a remote site to a local system and then executed on that local system without explicit action on the part of the user [53]. In the case of target classification, there may be several different classifiers available at a central site. Different classifiers

are suitable for different target types. A new classifier, which outperforms others for the given target type, can be downloaded to the local sensor node and replace the default classifier.

1.5 Agent Technologies

Since collaborative processing is essential in wireless sensor networks, a new computing paradigm is needed to facilitate the collaboration among sensor nodes. There are two kinds of computing paradigms: the client/server paradigm and the mobile agent paradigm, which we will discuss in detail in the next chapter. Here we briefly introduce the concept of agent and the mobile agent systems.

Simply put, agents are just independently executing programs which are capable of acting autonomously in the presence of expected and unexpected events. The central concept which distinguishes agents from simple programs is their interaction with their environment [71]. In another word, agents can be *embedded* or *situated* within their environments.

Agents can be defined by a set of attributes: active, autonomous, goal-driven, and typically acting on behalf of a user or another agent. The research of agent technology can be divided into two communities: the intelligent and multi-agent systems community, and the mobile agent community. Multi-agent systems are those in which an agent has the ability to be social and to interact with other agents, while mobile agent systems are those where an agent has the mobility feature, which enables modification of both services and customer profiles at separate locations.

1.5.1 Agent Standards

The Foundation for Intelligent Physical Agents (FIPA) [15] is a non-profit standards organization established in 1996 to promote the development of generic agent technologies that maximize interoperability within and across agent-based applications. Part of its function is to produce a specification for an agent-enabling software framework. Contributors are free to produce their own implementations of the software framework as long as its construction and operation

comply with the published FIPA specification. In this way, individual software frameworks can be interoperable. The FIPA agent standard aims to bring the commercial world a step closer to true software components, the benefits of which will include increased reusability, together with ease of upgrade.

1.5.2 Agent Communications

Communication enables agents to coordinate their actions and behavior, resulting in systems that are more coherent. Moreover, through coordination, agents can better achieve their design objectives or system's goals. Communication among agents has been most successfully modeled using the *speech act theory* [115]. The Knowledge Query Meta-Language, or KQML [49], was one of the first initiatives to specify how to support the social interaction characteristics of agents using a protocol based on speech acts. It involves three aspects: the method of message passing, the format, or syntax, of the information being transferred, and the meaning, or semantics, of the information. One of the main drawbacks with KQML is its lack of well-defined semantics, which forces the FIPA specification of its own agent communication language (ACL) [15]. The FIPA ACL describes a standard way to package messages, in a way that is clear to other agents what the purpose of the communication is.

1.5.3 Agent Migration

Migration means the movement of an agent to another location in the network and transparent continuation at the point before the migration occurs. That means code and state, such as data and execution information of the agent, must be transferred to and restored at the other location. Since transferring all these aspects is difficult and very expensive, we can distinguish between strong and weak migration. Strong migration means the transfer of the agent's code and its complete state, whereas weak migration can be defined as every migration that is not strong. The strong migration consists of two aspects: *code* and *state migration* [151]. The state

migration is further composed of *execution* and *data migration* [68] which means that the state of an agent is generally made up of the current execution point and the current data of the agent.

1.5.4 Mobile Agent Systems

The focus of this dissertation is the mobile-agent-based computing paradigm. Mobile agents are processes (i.e., executing programs) that can migrate from one machine of a system to another in order to satisfy requests made by their clients [72]. There are many mobile agent systems developed so far, of which six examples are briefly discussed.

Concordia is a mobile agent system developed at the Mitsubishi Electric ITA Laboratory in Waltham, Massachusetts [136]. It is a Java-based system that addresses security and reliability concerns. Concordia deploys an identity-based system protection of agents and it also relies on hashing the agent code, thereby protecting access to its resources and extensions to the Java security manager. Reliable network transmission is achieved using a message queuing subsystem based on a two-phase-commit protocol.

Agent TCL/D'Agents [56] is a mobile agent system created at Dartmouth College. It started as a Tcl/Tk based system, but later was extended to support Java, Scheme, and C/C++, with the new name, D'Agent. There are two types of agents in Agent TCL – those that move from machine to machine accessing resources, and those that remain on the machine whose purposes are to provide specific services not inherently provided by the system. Agent TCL has been used in both information-management and information retrieval applications.

Mole [28] is one of the first academic agent systems written in Java. There are different types of communication among agents: service-to-agent interaction, which is very much like the Remote Procedure Call (RPC) type client/server communication; session, which is the communication between mobile agents; anonymous group agent communication; and user agent communication. Mole supports asynchronous communication by an event-driven model. In

this model, depending on the input, internal rules, state information, and timeout intervals, the output events are generated, which in turn may be the inputs for other synchronization objects.

TACOMA [126] is a joint project between Tromso University in Norway and Cornell University in US. One of the applications using TACOMA system is the *WeatherStorm* distributed application. While other mobile agent research addresses programming language aspects of mobile agents, TACOMA mainly addresses operating system aspects. An agent needs to store code and data for future computations. It must be able to carry this information around when it migrates, and later retrieves it. Also, agents should be allowed to leave data behind at hosts or share data with other agents.

Voyager [7], developed by ObjectSpace, is one of the few systems that has achieved wide deployment. Goals of this product are to create state-of-the-art distributed programs quickly and easily, while providing a large degree of flexibility and extensibility for the products that are being created. It has many communication mechanisms, such as remote method invocation, object request broker, and DCOM support. It is a 100% Java-based system.

MASIF [88] is the first attempt to standardize the mobile agent system interoperability. It has been developed by IBM, General Magic, GMD Fokus, Crystaliz, and The Open Group. MASIF standardizes the interoperability between mobile agent systems, by specifying agent management, transfer, and naming. MASIF has been accepted as an OMG technology and reference implementations are being pursued by its proposers as well as by other companies.

1.6 Contributions

In this dissertation, we focus on the energy-efficient designs for CSIP in wireless sensor networks. The work has the following contributions:

Modeling, Evaluation and Implementation of Distributed Computing Paradigms. An energy-efficient, high-performance distributed computing paradigm, the mobile agent paradigm, is developed to support CSIP in sensor networks, where instead of each sensor node sending

local information to a processing center, as is typical in client/server-based computing, the processing code is moved to the sensor nodes through mobile agents. This approach has great potential in providing energy-efficient and scalable collaborative processing with low latency. We first present analytical models for both the client/server paradigm and the mobile agent paradigm, then performance evaluations are carried out through simulation. We employ the *execution time*, *energy* and *energy*delay* as metrics to measure the performance. Several experiments are designed to show the effect of different parameters on the performance of the paradigms. Experimental results show that the mobile-agent-based model does not always perform better than the client/server-based model. However, in the context of ad hoc sensor networks with hundreds or even thousands of nodes, unreliable communication links, and reduced bandwidth, the mobile-agent-based computing provides effective solutions for low network latency. From experiments, we also find out that the mobile agent model is less affected by network parameter changes; indeed, stability is one of its biggest advantages.

Cluster-based Hybrid Computing Paradigm. Based on the performance evaluation results, which show that the mobile agent paradigm performs much better when the number of nodes is large, while the client/server paradigm is more advantageous when the number of nodes is small, we then propose a cluster-based hybrid computing paradigm to combine the advantages of these two paradigms. There are in total four schemes in this paradigm, and simulation results show that there is always one scheme which performs better than either the client/server or the mobile agent paradigms. Thus, the cluster-based hybrid computing method provides an energy-efficient and high-performance solution to CSIP.

Mobile Agent Planning Modeling and Algorithms Design. The mobile agent migration route, including the selection of nodes and the order of migration, determines the amount of energy consumption, data fusion accuracy, and mobile agent migration time and thus has a significant impact on the overall performance of the sensor network. An improper design of

itinerary (or route) of mobile agent migration can largely deteriorate the performance of collaborative processing. As a result, one of the most challenging problems in mobile agent based computing, the design of mobile agent itinerary, is examined. We first present the mathematical models for both the static mobile agent planning and the dynamic mobile agent planning. We then propose three itinerary planning algorithms, the Information-driven Static Mobile Agent Planning (ISMAP), the Information-driven Dynamic Mobile Agent Planning (IDMAP), and the Predictive Information-driven Dynamic Mobile Agent Planning (P-IDMAP), with the goal of consuming a minimum amount of energy, spending the least number of hops, and prolonging the lifetime of the whole network. We develop several experiments to investigate the effect of different parameters on the performance of the algorithms. We design three metrics (*energy consumption*, *network lifetime*, and *the number of hops*) and use simulation tools to quantitatively measure the performance of different itinerary planning algorithms for collaborative processing. We show, through simulation, the advantages of the P-IDMAP algorithm in terms of energy consumption, network lifetime, and the number of hops.

Decentralized Reactive Clustering. In order to support the cluster-based hybrid computing and guide the mobile agent migration process, a clustering protocol is necessary. Different clustering protocols can affect the performance of the network to a great extent. Most existing clustering protocols either do not adequately address the energy-constraint problem or derive clusters proactively which may not be suitable for event-driven collaborative processing in sensor networks. We thus propose a decentralized reactive clustering (DRC) protocol where the clustering procedure is initiated only when events are detected. It has several desirable features: First, it is an event-driven protocol that forms clusters reactively. That is, only those nodes close to phenomena take part in the clustering procedure while other nodes can still be in the sleep mode. Thus the energy usage of the entire network can be reduced. Second, it employs power control techniques to lower the energy usage. It forms a two-tier hierarchy to support collaborative data processing. The performance of DRC is compared with another popular clustering

algorithm, LEACH [61]. Simulation results show considerable improvements over LEACH in energy conservation and network lifetime using the DRC approach.

1.7 Dissertation Outline

The remainder of this dissertation is organized as follows:

Chapter 2 first describes the mobile agent framework (MAF) and one application using MAF. It then presents a mathematical model of the client/server and the mobile agent paradigm. Three metrics are employed to evaluate the performance, and then a thorough simulation is carried out to determine under which conditions that one computing paradigm performs better than the other.

Chapter 3 presents a cluster-based hybrid computing paradigm, which is an enhancement over the client/server paradigm and the mobile agent paradigm. It divides the network into clusters. Four distinct schemes can be used according to the configuration of the clusters.

Chapter 4 discusses the mobile agent planning (MAP) problem. It first proposes the Information-driven Mobile Agent Migration for collaborative processing. Then it models the procedures of mobile agent migration for both static planning and dynamic planning. Three migration algorithms are proposed and performance evaluations are carried out.

Chapter 5 proposes a decentralized reactive clustering (DRC) protocol to support the cluster-based hybrid computing paradigm. It is a reactive protocol which is only invoked by events. It uses power-control techniques to reduce the energy consumption. It consists of the post-deployment phase, the cluster-forming phase, the intra-cluster data processing phase, and finally the cluster head-to-processing center phase.

Chapter 6 summarizes the breakthroughs of this work, and discusses possible future developments.

Chapter 2

Distributed Computing Paradigms Supporting CSIP: Modeling, Evaluation, and Implementation

In the previous chapter, we discussed the background knowledge of wireless sensor networks and collaborative signal and information processing (CSIP), as well as the computing paradigms that support CSIP. In this chapter, the focus is on the development of distributed computing paradigms to support CSIP in wireless sensor networks to combat the unique challenges in sensor networks. We will further discuss the modeling, performance evaluation of the client/server and mobile agent paradigms, and the implementation of the mobile agent paradigm. Seven experiments are designed for performance evaluation under varying network conditions.

2.1 Computing Paradigms

In the context of wireless sensor networks, the computing paradigm refers to the information processing model deployed at the application layer of the protocol stack. Since collaborative

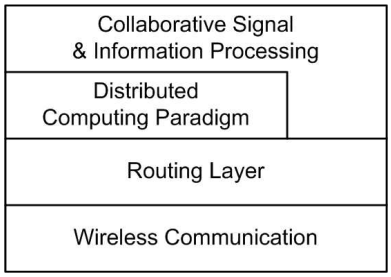
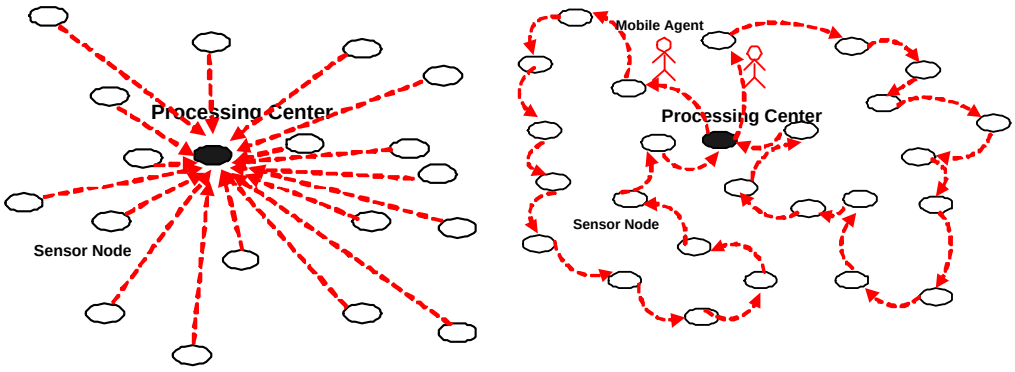


Figure 2.1: Protocol stack of wireless sensor networks.

processing is essential in wireless sensor networks, a computing paradigm is needed to facilitate the collaboration between sensor nodes. As seen from Fig. 2.1, it is located above the routing layer and is used to support the collaborative signal and information processing. There are two types of computing paradigms for use: the client/server paradigm and the mobile agent paradigm.

The client/server paradigm has been one of the most popular models adopted in distributed computing [51, 73]. In this paradigm, a server offers a set of services, resources, and know-how needed for service execution. The client requests the execution of a service. As a response, the server performs the requested service by executing the corresponding service know-how and accessing the required resources at the server. In wireless sensor networks, the server is the processing center and the clients are the sensor nodes. Fig. 2.2(a) illustrates the client/server-based paradigm. Various systems are built based on the client/server paradigms in wireless sensor networks. The habitat monitoring system [86] is one example. It uses the client/server computing paradigm to transfer data from sensor nodes to the gateway to implement habitat monitoring functions. Another example is the IVY system [5], a wireless sensor network infrastructure used at Berkeley, that adopts a multi-level client/server paradigm.



(a) The client/server computing paradigm.

(b) The mobile-agent computing paradigm.

Figure 2.2: Different computing paradigms.

Although widely used, the disadvantages of this model are also dramatic, especially for wireless sensor networks [63, 98, 127]. First of all, in this model, there should be some super-nodes acting as processing centers, which need much higher energy, storage and computing capabilities. These will largely reduce the lifetime of the whole wireless sensor network, especially in autonomic and homogeneous sensor networks. Secondly, in data fusion, large amounts of data gathered by the sensor nodes have to be moved from the clients to the server. In a system with many clients, the total bandwidth requirements may exceed the actual available bandwidth, resulting in poor performance of the system. Finally, the topology of wireless sensor networks determines that the closer a node is to the server, the higher energy the node will consume because it has to be an intermediate node to route the packets from other nodes. This results in nodes closer to the server dying much more quickly than other nodes.

The mobile-agent-based paradigm [64, 87, 90, 110, 137] is an elegant alternative. In this paradigm, the service know-how is owned by the server, which dispatches the mobile agents, but most of the resources are located at the clients. The server sends out mobile agents carrying the service know-how. The mobile agents complete the service using resources available at the

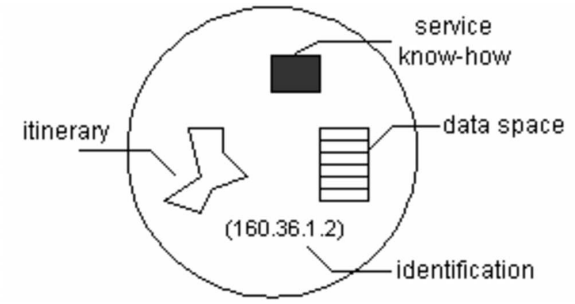
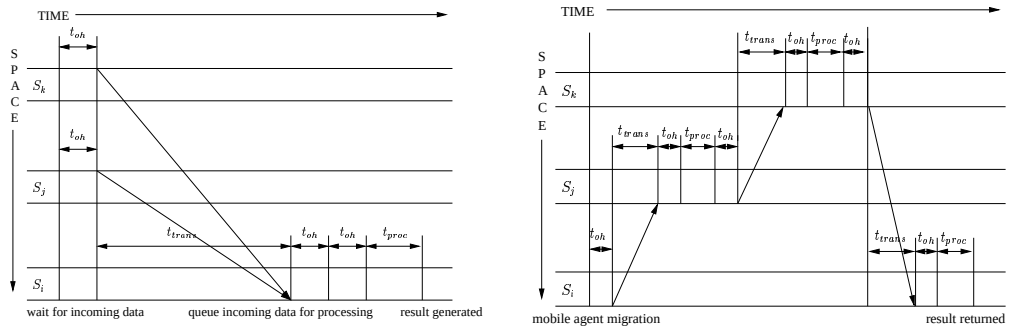


Figure 2.3: Mobile agent components [99].

clients. In wireless sensor networks, it is the processing center that sends out the mobile agents and finally receives the mobile agents. The mobile agent will migrate to each node and conduct data fusion. It then carries the partially integrated results and continues migration. Fig. 2.2(b) shows the framework of the mobile agent model.

Generally speaking, a mobile agent is a special kind of software that can execute autonomously. We define the mobile agent as an entity of four attributes: identification, itinerary, data space, and service know-how, as shown in Fig. 2.3 [99]. Identification uniquely specifies each mobile agent. Data space is the agent's data buffer which carries a partially integrated result. Itinerary is the route of migration. It can be fixed or dynamically determined based on the current network status. Service know-how is the processing task (or execution code) carried with the agent.

In order to better illustrate how these two computing models perform, we present a temporal and spatial comparison of the life cycle of their migration units in [101, 140]. Here, we briefly summarize the comparison in Fig. 2.4. In the client/server-based model, the migration unit is "data", while in the mobile-agent-based model, the migration unit is "mobile agent". The life cycle of both units is composed of three components: t_{trans}, the time spent in transferring the migration unit from one node to the other, t_{oh}, the overhead time and t_{proc}, the processing time. In the case of client/server-based model, t_{oh} is the time spent on file access. Here, we assume



(a) Data migration.

(b) Mobile agent migration.

Figure 2.4: Life cycle of data (client/server-based) and mobile agent (mobile-agent-based) migration in time and space.

that data collected by sensor nodes is stored in a file; in the case of mobile-agent-based model, in addition to the time spent on file access, it also includes the time used to create, dispatch, and receive the mobile agent, which makes the overhead time larger than that in the client/server-based model. Note that the length of t_{trans}, t_{oh}, and t_{proc} are used here as symbols to show the sequence of events in the life cycle. They do not reflect the actual time spent.

In the client/server-based model (Fig. 2.4 (a)), if we assume S_j and S_k are identical nodes and can start data transfer at the same time, then the clients spend t_{oh} to read the data files and t_{trans} to transfer them to the server (S_i). S_i uses t_{oh} amount of time to receive each data file and access it. After receiving all the incoming data files, the server can start processing, which would take t_{proc} amount of time.

In the mobile-agent-based model (Fig. 2.4 (b)), a mobile agent is created at node S_i using t_{oh}. It then migrates to S_j using t_{trans}. S_j spends t_{oh} to parse the mobile agent and read the data file and t_{proc} to do data fusion. After an amount of time t_{oh}, the agent is sent from node S_j to S_k. After the same procedure, it finishes migration and returns to S_i.

The mobile agent model has many advantages over the client/server model [59, 77, 94, 98] and is especially suitable for distributed computing in wireless sensor networks:

(1) Performance. Network bandwidth requirement is reduced for the mobile-agent-based computing. Instead of passing large amounts of data over the network through several round trips, only the agent (with a relatively small size) is sent. This is especially important for real-time applications and where the communication is through low-bandwidth wireless connections.

(2) Energy. Since the total amount of data transmission is reduced, the energy usage can also be reduced, because a major part of the energy consumption goes to radio transmission [47]. Furthermore, in client/server-based computing, there are generally some super-nodes acting as processing centers, which need much higher energy, storage and computing capabilities. These will usually reduce the lifetime of the whole wireless sensor network, especially in autonomic and homogeneous sensor networks. The mobile-agent-based computing paradigm, on the contrary, distributes the energy usage much more evenly among all the nodes and processing center, so the lifetime of the wireless sensor network can be prolonged. Moreover, the mobile agent can terminate migration and return when the accuracy of the fusion results satisfy the requirement, thus further reducing the energy usage and execution time since unnecessary data transfers are avoided.

(3) Scalability. For client/server-based computing, there will be increased queuing length as the number of clients increases. As a result, it may cause longer processing times and more possible drops at the server side. Unfortunately, in wireless sensor networks, the number of nodes may reach hundreds or even thousands. On the other hand, the mobile-agent-based computing may not be significantly affected as the number of nodes increase.

(4) Reliability. When a node is down, which occurs quite often in sensor networks, the mobile agent can bypass that node and migrates to the next available node. Therefore, the performance of mobile agent is not appreciably affected by the reliability of the network.

2.2 Evaluation Metrics

Even though the mobile-agent-based computing method is promising and seems more appropriate for wireless sensor networks, we need to determine the key factors in wireless sensor networks that affect the performance of mobile-agent computing. The mobile-agent-based computing scheme may not always perform better than the client/server-based computing since mobile agents also introduce overhead, which not only comes from the agent creation and dispatch time but also from file accesses. On the other hand, the client/server-based computing methodology needs to transfer data files to the processing center, which also causes overhead due to file accesses. Therefore, a performance evaluation of these two computing paradigms under different scenarios is essential. In [140], we reported some preliminary simulation results using GloMoSim [150], which only uses the execution time as evaluation metric. In [101], an energy model is derived through analytical modeling. In this research, we employ three metrics to evaluate the performance. The lack of energy modeling capability prevents us from using GloMoSim. After careful study of current existing simulation software, we chose to use Network Simulator 2 (ns-2) [30] to simulate the wireless sensor network. ns-2 is a discrete event simulator targeted at networking research and is the most popular choice of simulators used in academia. Following the OSI seven layer network architecture, ns-2 divides a network into an application layer, transport layer, network layer, MAC layer, physical layer, and radio and mobility layer. In each layer, ns-2 provides different models and protocols. In the following subsections, the three performance metrics are described in detail.

2.2.1 Assumptions

For performance evaluation purposes, we make the following assumptions in our simulations:

- When a certain event occurs in the sensor field, all sensor nodes can detect it and collect the raw data. This is to ensure that there are always the same number of sensor nodes participating in the data fusion.
- There are no events simultaneously occurring in the field.

2.2.2 The Execution Time Metric

The *execution time* is the time spent to finish an information processing task. In the mobile-agent-based paradigm, it starts from the time a mobile agent is created to the time the mobile agent returns with results. In the client/server-based paradigm, it is from the time the clients send out requests to the time the server generates results. The execution time consists of three parts – data transfer time (t_{trans}), overhead time (t_{oh}), and data process time (t_{proc}). In [101], we found that in both client/server-based computing and mobile-agent-based computing, we have exactly the same amount of data to process and the only difference is where data processing takes place. Therefore, the data process time (t_{proc}) is the same for both computing paradigms and we need not include this part in our model.

For the client/server paradigm, the execution time is determined by the network transfer rate v_n, the data file size s_f (the size of the raw data each node collects), the overhead of file access o_f (the time used to read and write a data file), the number of sensor nodes p, the number of agents m and the number of sensor nodes n that each agent migrates (the server is not included). Notice that $p = m \times n$. Thus, the data transfer time is $t_{trans} = mns_f/v_n$ and the overhead time is $t_{oh} = 2mno_f$ (assuming the time used to read and write the data file is the same). Therefore,

the total execution time for the client/server paradigm is:

$$t_{cs} = \frac{mns_f}{v_n} + 2mno_f. \tag{2.1}$$

For the mobile agent paradigm, the execution time is determined by the network transfer rate v_n, the mobile agent size s_a, the overhead of mobile agent o_a, the number of sensor nodes p, number of agents m, and the number of sensor nodes n that each agent migrates. Again, $p = m \times n$. Thus, the time used to transfer agents is $t_{trans} = (m+n)s_a/v_n$, since mobile agents use ns_a/v_n to transfer simultaneously and they return to server after finishing the task serially, which takes ms_a/v_n additional time for m mobile agents; the agent overhead time is $t_{oh} = 2(m+n)o_a$, since it takes $2mo_a$ for the server to send and receive m mobile agents in sequence, and $2no_a$ for the nodes to send and receive each mobile agent simultaneously. Therefore, the total execution time for the mobile agent paradigm is:

$$t_{ma} = \frac{(m+n)s_a}{v_n} + 2(m+n)o_a. \tag{2.2}$$

The mathematical model derived above characterizes the behavior of these two computing paradigms to some extent, but it is difficult to obtain any quantitative measurement, especially when modeling the data transfer time, where retransmission and error control are not considered. Unfortunately, these factors occur quite often in wireless sensor networks because of the use of wireless links. Therefore, we propose to use simulation tools for more accurate estimation of the data transfer time t_{trans}. The model used to calculate the execution time is then:

$$t_{cs} = t_{trans-cs} + 2mno_f, \tag{2.3}$$

and

$$t_{ma} = t_{trans-ma} + 2(m+n)o_a \tag{2.4}$$

where $t_{trans-cs}$ and $t_{trans-ma}$ are obtained from simulation.

2.2.3 The Energy Metric

The *energy* is another important metric to measure the performance of different computing paradigms in wireless sensor networks, since sensor nodes are energy-constrained. The energy metric measures the total energy the wireless sensor network consumes to finish a processing task. Similar to the execution time metric, the energy metric is also composed of three parts – data transfer energy (e_{trans}), overhead energy (e_{oh}), and data process energy (e_{proc}). Since in both cases there are same amount of data to process, we may safely ignore the data process energy (e_{proc}). For the same reason, we do not include the energy used for sensing either. From [111] the estimate of energy consumed during processing is shown with the following set of equations:

$$E_{consumed} = \int_{t_0}^{t_0+T_{exec}} P(t)dt, \tag{2.5}$$

where T_{exec} is the time taken for executing software and $P(t)$ is the instantaneous power of the processor. Also,

$$P_{average} = \frac{1}{T_{exec}} \int_{t_0}^{t_0+T_{exec}} P(t)dt. \tag{2.6}$$

which implies that

$$E_{consumed} = T_{exec} \times P_{average}. \tag{2.7}$$

Thus the overhead energy for one transmission can be computed by $2 \times p_{processor} \times t_{oh}$, where $p_{processor}$ is the power consumption of the node processor in full load, t_{oh} is either the overhead of file access o_f or the overhead of mobile agent o_a, the multiplier 2 considers both transmission and receiving that use the same amount of energy in the overhead processing. We choose $p_{processor} = 75$mW to simulate the ARM Thumb processor clocked at 40 MHz [47]. There are a total of $m \times n$ transmissions occurring in the client/server-based computing and $m \times (n+1)$ transmissions occur in the mobile-agent-based computing. For the data transfer energy (e_{trans})

part, we again use simulation tools for more accurate estimation. Therefore, the model used to calculate the energy usage is:

$$e_{cs} = e_{trans-cs} + 2mnp_{processor}o_f, \tag{2.8}$$

and

$$e_{ma} = e_{trans-ma} + 2m(n+1)p_{processor}o_a, \tag{2.9}$$

where $e_{trans-cs}$ and $e_{trans-ma}$ are obtained from simulation.

2.2.4 The energy*delay Metric

In wireless sensor networks, it is important to consider both energy and execution time. Clearly, a more suitable metric in wireless sensor networks is the combined effect, the *energy*delay* metric [81], which reflects both the energy usage and the execution time. We adopt the following equations:

$$energy * delay_{cs} = e_{cs} \times t_{cs}, \tag{2.10}$$

and

$$energy * delay_{ma} = e_{ma} \times t_{ma}, \tag{2.11}$$

where e_{cs} and e_{ma} are the energy usage, and t_{cs} and t_{ma} are the execution time of the client/server and the mobile agent paradigms respectively.

2.2.5 Experimental Parameter Setup

In our simulation, the basic network consists of wireless nodes randomly distributed within a 10 m by 10 m square area. The mobility model we use is the random waypoint model, where nodes randomly choose a destination and move at a speed of 1 m/s. Once the nodes arrive at the destination, they pause for 2 seconds and then continue moving. Other parameters that

Table 2.1: Parameters for the basic network setup.

Network area		10×10 m^2
Node placement		Random
Initial energy		36 Joules
Mobility		Random waypoint, 1 m/s with 2 s pause
Transmission range	Radio frequency	914 MHz
	Transmission power	0.6 W
	Receiving threshold	3.652×10^{-10} W
Propagation model		Two-ray ground reflection
MAC layer		IEEE 802.11
Routing protocol		DSDV
Transport layer protocol		UDP

characterize the network include the MAC layer protocol, the routing protocol, the transport layer protocol, the transmission and receive power, the propagation model, radio frequency, and radio receiving threshold. The default parameter values in the basic network setup are listed in Table 2.1.

Parameters that affect the data processing time (t_{proc}) and the overhead (t_{oh}) in different computing paradigms include the number of nodes (p) [in the mobile-agent based paradigm, p is the multiplication of the number of mobile agents (m) and the number of nodes migrated by each agent (n)], the data file size (s_f), the mobile agent size (s_a), the mobile agent overhead (o_a), the file access overhead (o_f), the network transfer rate (v_n), and the data processing rate (v_d). The parameter values in the basic network setup are shown in Table 2.2.

2.3 Experiments and Simulation Results

We designed six experiments to evaluate the effect of different parameters on the execution time, energy usage and energy*delay. These parameters include the number of nodes, the number of mobile agents, the ratio between data size and mobile agent size, the overhead ratio, the

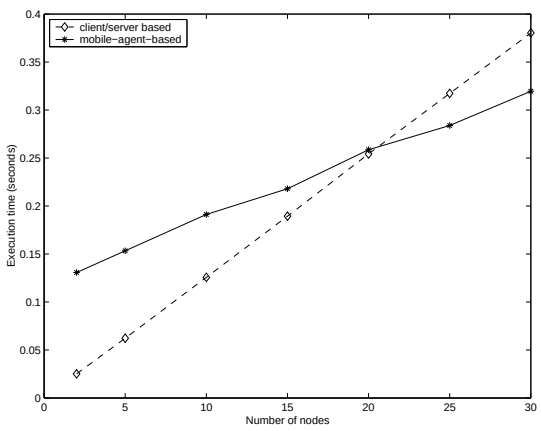
Table 2.2: Computing paradigm related parameters for the basic network.

n	m	s_f	s_a	o_f	o_a	v_n	v_d
20	1	1000 b	200 b	0.00005 s	0.0002 s	2 Mbps	100 Mbps

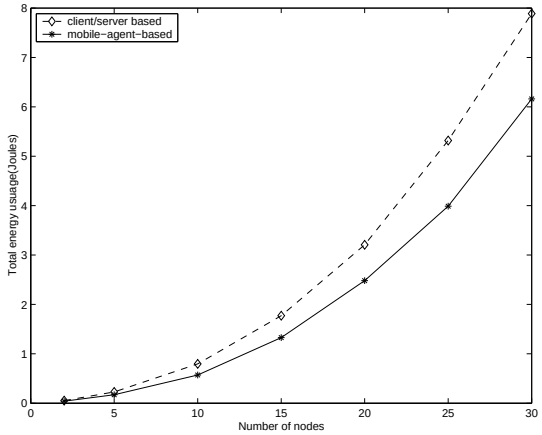
node transmission range, and different underlying protocols or models in each layer. In each experiment, we changed one of the parameters of the basic network and kept all the others unchanged. We conducted five simulations and calculate their means and standard deviations. Since the standard deviations are less than 2% of the means, only the means are plotted in the figures.

2.3.1 Effect of the Number of Nodes (p)

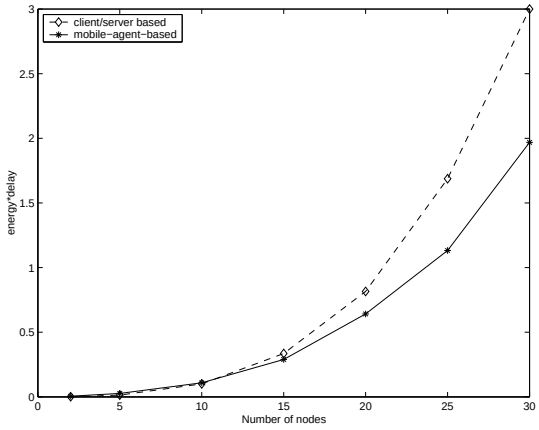
In this experiment, we changed the number of nodes p from 2 to 30. The execution time measurement is shown in Fig. 2.5(a). It can be observed that the execution time using both paradigms grows as the number of nodes increases, but the client/server model grows much faster than the mobile-agent model. This is because as the number of nodes increases, the server has to deal with more connections requested by the clients at the same time, which increases the execution time. On the other hand, the mobile agent model is less influenced by the number of nodes because there are much fewer connections at any given time for the mobile agent model. The figure also shows that, for $p < 20$, the client/server model performs a little better than the mobile agent model. This happens because the mobile agent model needs more connections than the client/server model in order to send and receive mobile agents. Another reason is that the overhead of the mobile agent surpasses the overhead of the client/server model. Therefore, in a network with fewer nodes, the client/server model may have a shorter execution time than that of the mobile agent model. However, if the number of nodes is large, the mobile agent model will perform better. As for the total energy the network consumed shown in Fig. 2.5(b),



(a) t_{cs} vs. t_{ma}.



(b) e_{cs} vs. e_{ma}.



(c) energy*delay.

Figure 2.5: The effect of the number of nodes (p).

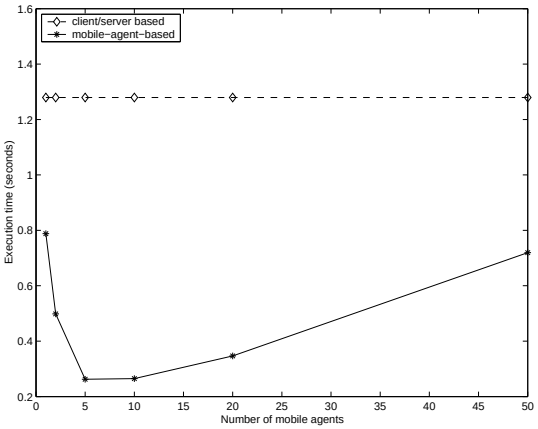
mobile-agent computing almost always consumes less energy, because the amount of data transmitted are significantly reduced in the mobile agent computing, thus saving total energy. The pattern is also seen in the energy*delay profile shown in Fig. 2.5(c), that the mobile agent model quickly shows an advantage over the client/server model when $p > 10$.

2.3.2 Effect of the Number of Mobile Agents (m)

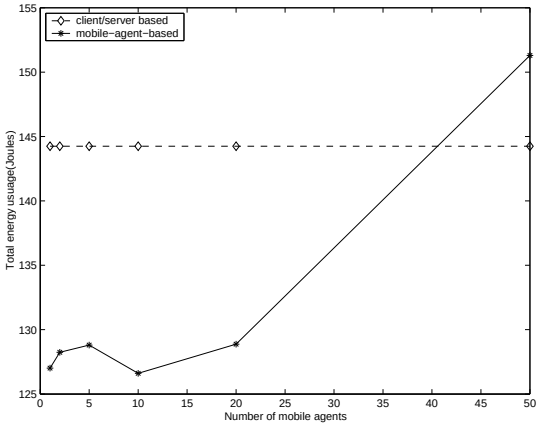
In this experiment, we fixed the node number at 100 and observed the effect of the number of mobile agents on the performance of different computing paradigms. Without loss of generality, we assume that each agent migrates the same number of nodes. We expect a constant profile from the client/server-based computing since it is irrelevant to the number of mobile agents. We observe from Fig. 2.6(a) that the mobile agent model always has less execution time than the client/server model because the node number is large. Interestingly, the execution time of the mobile agent model decreases as the number of mobile agents increases and reaches the lowest point when there are five mobile agents. Above that, the execution time begins to increase. This is because more mobile agents will reduce the number of nodes each agent migrates, thus reducing the execution time. However, using more mobile agents also cause more connections and overhead. Therefore, a proper number of mobile agents can make the model perform more efficiently. The energy*delay metric in Fig. 2.6(c) also shows that when $m = 10$ it has the lowest energy*delay value.

2.3.3 Effect of the Data Size/Mobile Agent Size (s_f/s_a)

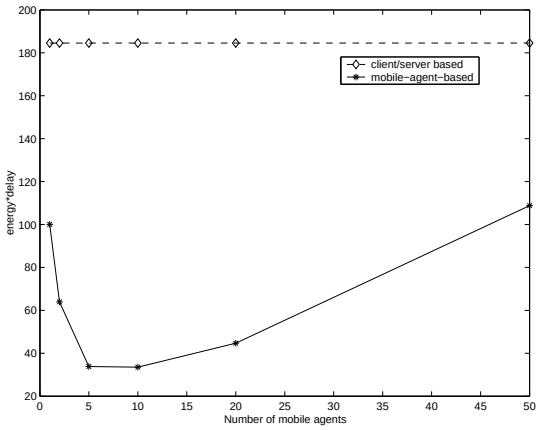
In this experiment, we changed the size of the data file s_f, but fixed the other parameters and let $s_a = 200$ b, p=20. Fig. 2.7(a) shows the result. We observe from Fig. 2.7(a) and Fig. 2.7(b) that the execution time and energy consumption using the mobile-agent based computing are constant because data are located at the local nodes; only a fixed amount of results are transferred. When the data size is less than 1.1 kb, the client/server-based computing method has



(a) t_{cs} vs. t_{ma}.

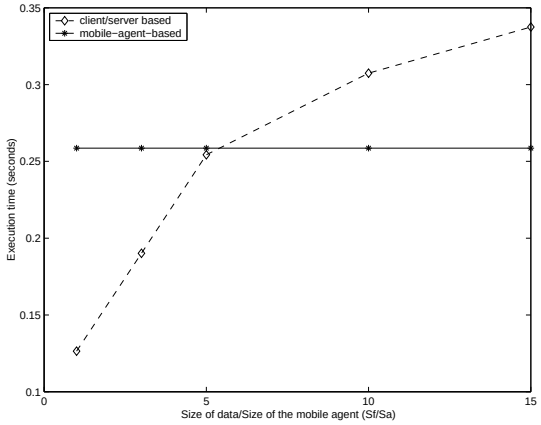


(b) e_{cs} vs. e_{ma}.

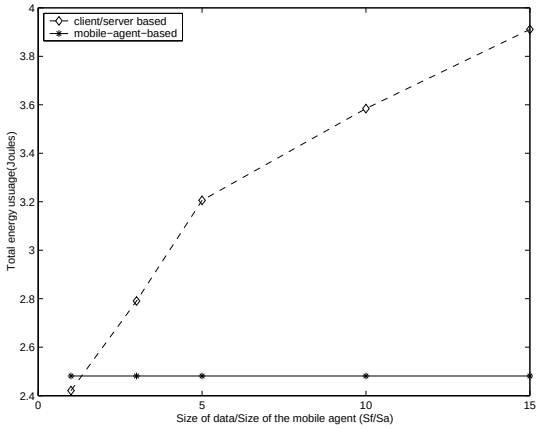


(c) energy*delay.

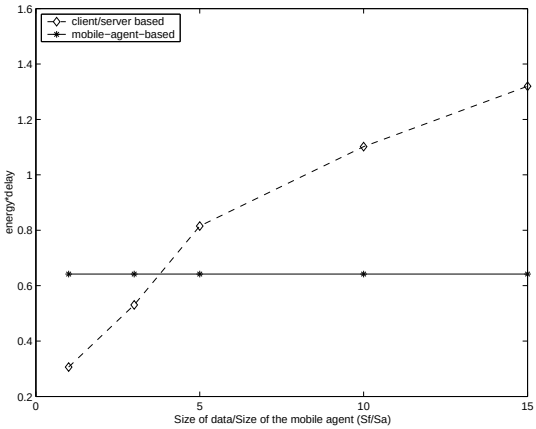
Figure 2.6: The effect of the number of mobile agents (m).



(a) t_{cs} vs. t_{ma}.



(b) e_{cs} vs. e_{ma}.



(c) energy*delay.

Figure 2.7: The effect of data size vs. mobile agent size (s_f / s_a).

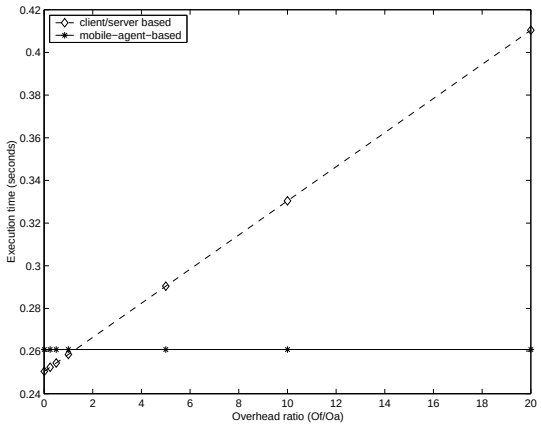
less execution time than that of the mobile-agent case. However, the larger the data size, the more advantageous the mobile-agent based paradigm is. This is because as s_f increases, more data needs to be transferred, thus increasing the time used by the client/server model. For the same reason, when the data size is less than 300 b, the client/server-based computing format consumes less energy.

2.3.4 Effect of the Overhead Ratio (o_f/o_a)

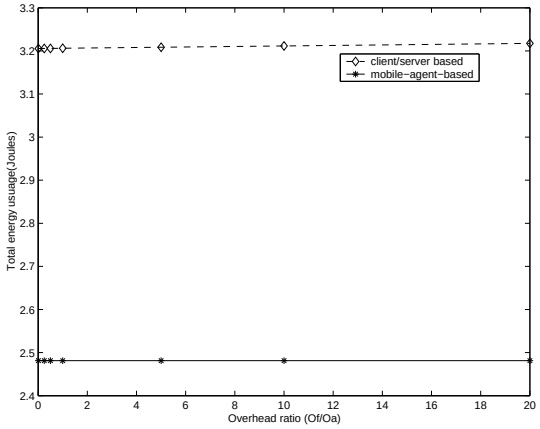
In this experiment, we fixed all other parameters in the basic network, and observed the effect of the overhead ratio o_f/o_a (between 0.01 and 20.0) to the performance of different computing paradigms. We can see from Fig. 2.8(a) that when the ratio is greater than 1.5, the client/server-based computing starts to perform worse than the mobile-agent model since the larger the o_f, the longer the execution time. From Fig. 2.8(b) we observe that the client/server paradigm always consumes more energy. Fig. 2.8(c) gives the energy*delay metric, which shows the combined effects of energy usage and execution time.

2.3.5 Effect of the Node Transmission Range

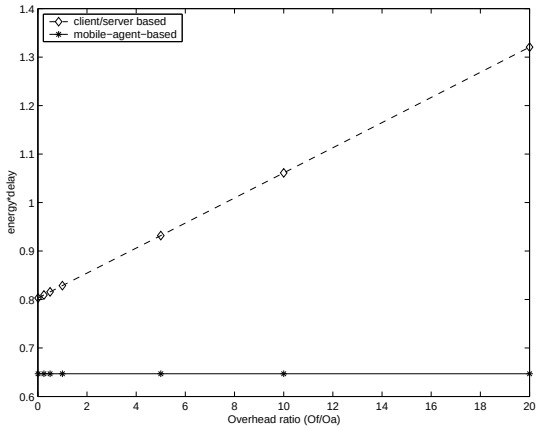
In this experiment, we fixed all other parameters in the basic network, but changed the transmission range from 49 m to 198 m. Fig. 2.9 shows the effect of transmission range. The execution time remains the same while energy consumption decreases as the transmission range increases. The explanation is straightforward. Since the sensor nodes are deployed in a 10 m by 10 m field, the transmission range of sensor nodes is always large enough to connect each other through a single hop. Thus, the execution time is not affected by the changes of the transmission range. However, as the transmission range increases, more transmission power is needed, thus consuming more energy. Again, the mobile agent model is less affected by the change of transmission range than is the client/server model. The energy*delay graph also shows similar patterns.



(a) t_{cs} vs. t_{ma}.

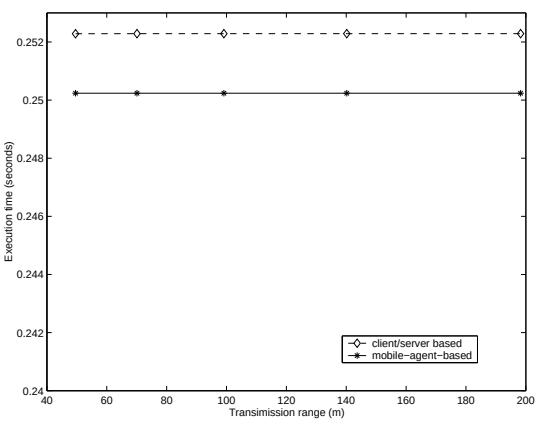


(b) e_{cs} vs. e_{ma}.

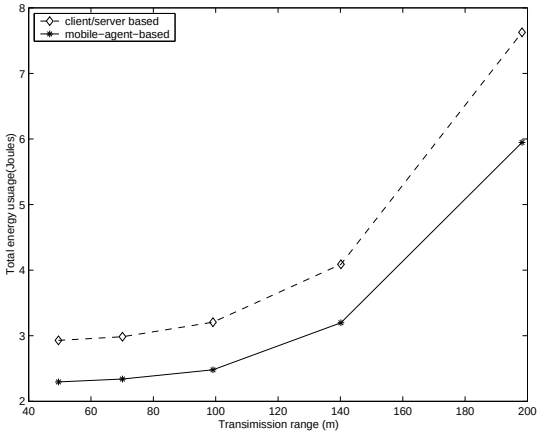


(c) energy*delay.

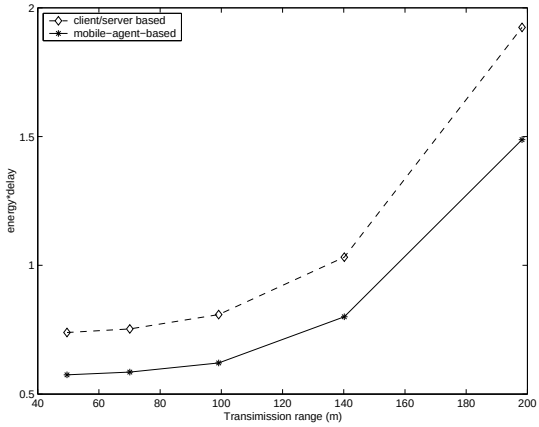
Figure 2.8: The effect of the overhead ratio (o_f/o_a).



(a) t_{cs} vs. t_{ma}.



(b) e_{cs} vs. e_{ma}.



(c) energy*delay.

Figure 2.9: The effect of the transmission range.

2.3.6 Effect of Different Protocols and Models

In this experiment, we changed the protocols and models used at different layers of the network, but fix the other parameters in the basic network and kept $p = 20$.

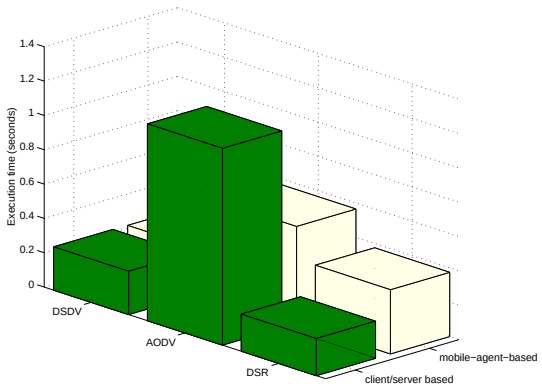
Fig. 2.10 shows the effect of different routing protocols, including AODV, DSDV and DSR. It can be seen from Fig. 2.10(a) and Fig. 2.10(b) that both the client/server and mobile-agent computing perform the best using the DSR protocol in terms of the execution time and energy consumption. Fig. 2.10(c) shows the energy*delay values.

Fig. 2.11 demonstrates that in the physical layer, the different propagation models have little effect on the execution time, energy consumption, and energy*delay.

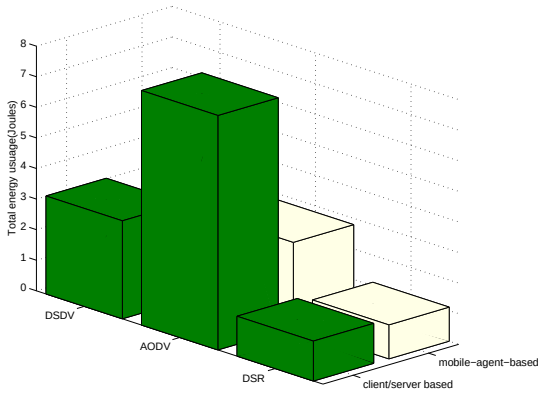
2.4 Detection Performance Comparison for Different Computing Paradigms

One of the main objectives of a wireless sensor network is to detect and classify events of interest, for instance the possible presence and type of an intruder. Therefore, the detection performance is one of the major performance metrics to evaluate a classifier. We now compare the detection performance using different computing paradigms in this section. The detection performance is characterized by the receiver operating characteristic (ROC) curve, which gives the relationship between the probability of false alarms and probability of successful detections. For both false alarms and successful detections, the detection signal exceeds its threshold, therefore concluding the presence of a target. If a target does exist in the area under monitoring, then we say that the target is successfully detected or it is a *true positive*. Otherwise, the detection decision is a false alarm or a *false positive*. If there is no target existing in the area, while the classifier reports no target, we call it a *true negative*, otherwise we call it a *false negative*.

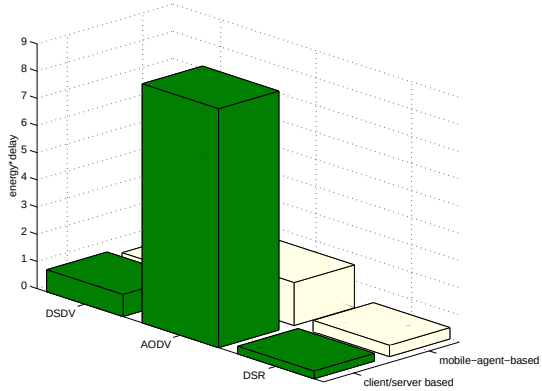
Each sensor has a particular ROC curve determined by the characteristic of the type of the sensor. For the acoustic amplitude sensor, the true positive rate is related to the distance to the



(a) t_{cs} vs. t_{ma}.

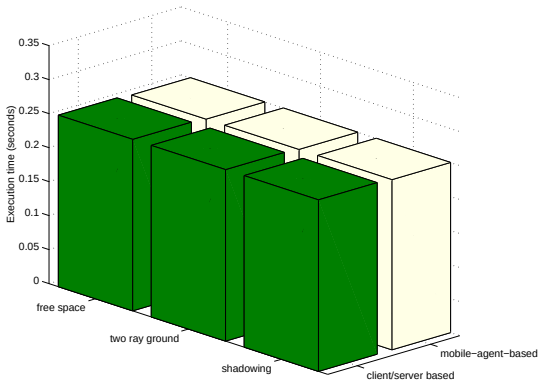


(b) e_{cs} vs. e_{ma}.

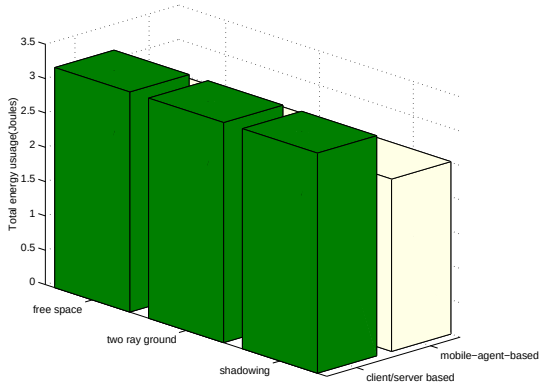


(c) energy*delay.

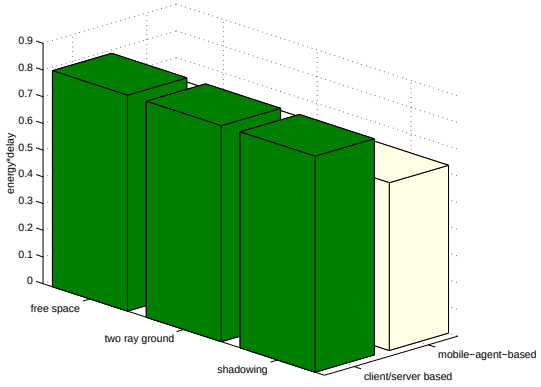
Figure 2.10: The effect of the different routing protocols.



(a) t_{cs} vs. t_{ma}.



(b) e_{cs} vs. e_{ma}.



(c) energy*delay.

Figure 2.11: The effect of the different propagation models.

Table 2.3: Simulation results for the client/server-based classifier.

Number of Simulations	50
Number of Correct	34
Accuracy	68.0%
Positive Cases Missed	4
Negative Cases Missed	12
Fitted ROC Area	0.874

target and is modeled by the following equation:

$$TP = \frac{maxDist - targetDist}{maxDist}, \tag{2.12}$$

where $maxDist$ is the maximal distance that a sensor can detect the target, $targetDist$ is the distance from the current node to the target. The true positive rate on a sensor node is the local detection decision. It is necessary to fuse the different detection decisions on different sensor nodes in order to get a more accurate detection result. We employ two kind of classifiers: the client/server-based classifier fuses the local decisions on a central node through the client/server computing paradigm; the mobile agent-based classifier uses a mobile agent to fuse the local decisions. The distributed data fusion algorithm the mobile agent employs is the modified MRI algorithm [101]. We developed a simulator in Java to simulate the process of data fusion for these two classifiers. In the simulation, 100 sensor nodes are randomly deployed in a 30×30 m^2 area. The ROC curve for a single sensor is the green dashed curve on Figure 2.12. We ran the simulation several times, including the situations that there is a target or no target. We then used a web-based calculator [8] to draw the ROC curve. Table 2.3 is the summary of the simulation results for the client/server-based classifier.

For the mobile-agent-based classifier, we allow the mobile agent to migrate among the nodes that are closer to the target and get a final integrated results. Table 2.4 is the summary of the simulation results for the mobile-agent-based classifier.

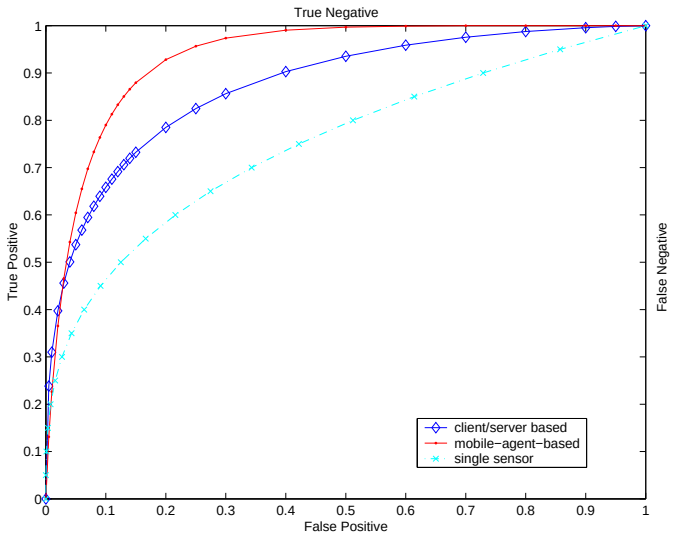


Figure 2.12: ROC curves for different classifiers.

Table 2.4: Simulation results for the mobile-agent-based classifier.

Number of Simulations	47
Number of Correct	40
Accuracy	85.1%
Positive Cases Missed	0
Negative Cases Missed	7
Fitted ROC Area	0.935

The ROC curves for the single sensor (plotted in green dashed line), client/server-based classifier (plotted in blue solid line), and the mobile-agent-based classifier (plotted in red solid line) are shown in Figure 2.12.

Area under an ROC curve (AUC) is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance [48]. From Figure 2.12, we find the ROC curve for single sensor has the smallest AUC, which indicates that the single-sensor classifier performance is worse than for a classifier utilizing data-fusion techniques. Furthermore, the ROC for the mobile-agent-based classifier has a bigger AUC (0.935) than that of the client/server-based classifier (0.874), we can conclude that it performs better than the client/server-based computing paradigm. The reason is that in the mobile-agent-based classifier, the mobile agent selectively migrates to the sensor nodes with higher local detection accuracy; thus, the final fused result is better than the client/server-based classifier, which fuses the results from all sensor nodes. To summarize, our simulations illustrate the following:

(1) In general, decision fusion can significantly improve the detection performance of distributed sensors.

(2) Better detection performance can be achieved by utilizing the mobile agent to selectively migrate the nodes with more accurate local detection results.

2.5 Discussions

From the experiments, we can conclude that under the following conditions the mobile agent computing paradigm performs better: when the number of nodes is large; when the ratio between data size and mobile agent size is large; when the overhead ratio between the data file and the mobile agent is large. We also show that the mobile agent based classifier for target detection in sensor networks has a better detection performance than the client/server based classifier. Thus, problems in sensor networks, with their hundreds or even thousands of nodes,

and related high node-failure rates, unreliable communication links, and reduced bandwidths, can be addressed very satisfactorily using the mobile agent computing paradigm.

2.6 A Mobile Agent Framework for CSIP

A mobile agent framework (MAF) suitable for CSIP in wireless sensor networks was developed by the University of Tennessee. The MAF is implemented in Python with interfaces to C++ routines. The Python language is chosen for its ability of *object-serialization*, which was used to generate the agent. The MAF was first implemented in an Intel machine environment, then cross-compiled using the SH-4 cross-compilers to generate an SH-4 version. The Python interpreter was also cross-compiled into SH-4 version. The MAF can be used in various CSIP applications, such as target classification, target localization and target tracking in battlefields. The MAF consists of a mobile agent daemon or server that runs on each node and acts as a host to the incoming *mobile agent*. The mobile-agent itself is a program that flies from one node to another, collecting and integrating local results from each node. The agent is created when an event is detected. Once a mobile agent daemon receives a mobile agent, the execution code the mobile agent carries is executed to process the data locally and a partial integration result is generated. The mobile agent then calls Direct Diffusion, which calls Sensoria RF modem API to communicate with other nodes. Fig. 2.13 shows the stages in the implementation of the MAF.

The MAF is useful in supporting CSIP in wireless sensor networks. Qi et al. [100] give a detailed description of the use of MAF in target classification in wireless sensor networks. Each sensor node senses interesting events and collects the corresponding signals. The signals are then processed to extract a feature vector, which is then classified using the k-nearest-neighbor (kNN) algorithm. Next, a *confidence level* for each class is generated by letting the k equal to several different values in the kNN. Suppose there are n different targets (classes) in the dataset. According to kNN, for each sample with unknown class, the algorithm looks into a

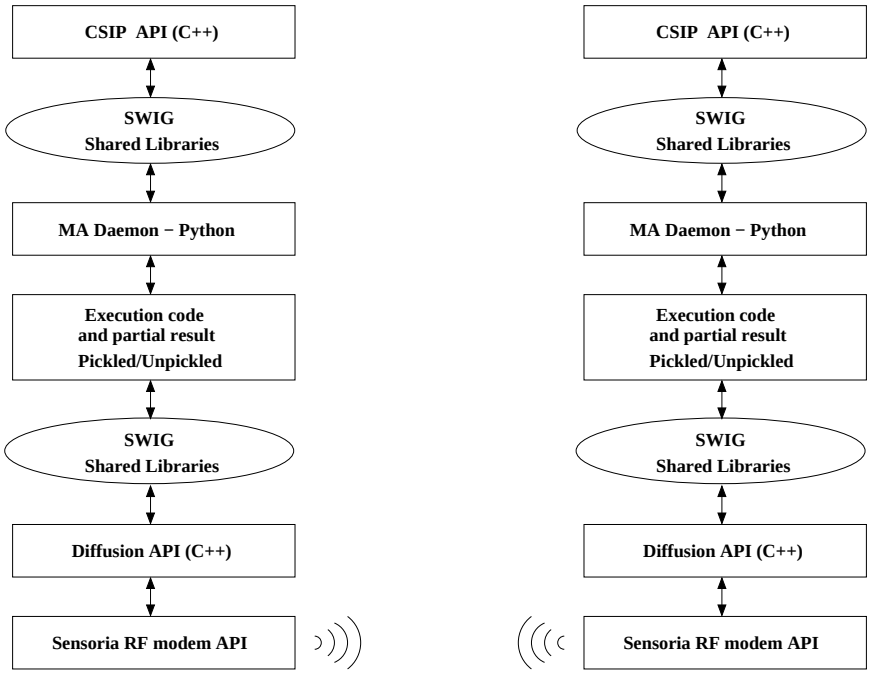
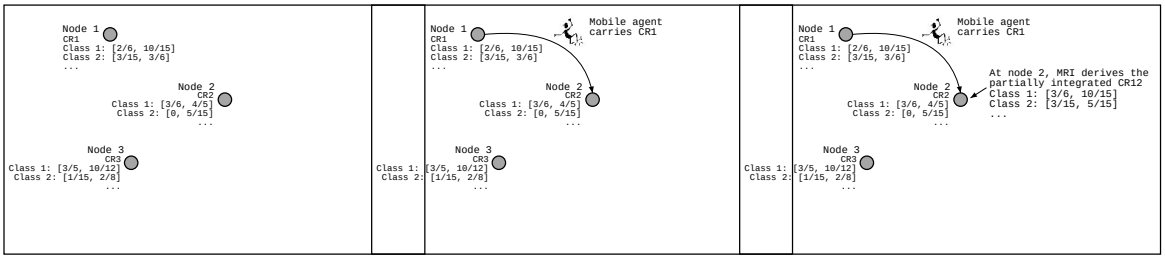


Figure 2.13: Implementation of MAF [76].

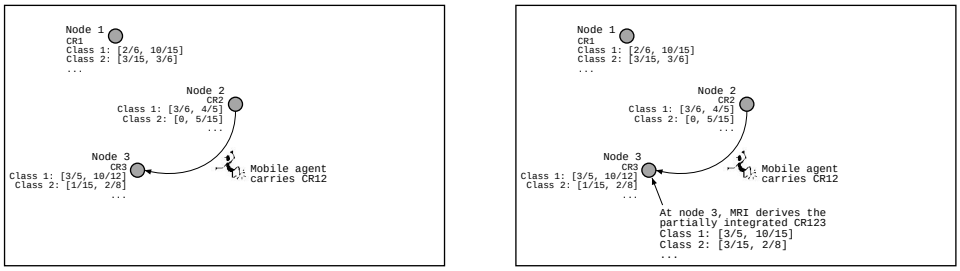
neighborhood of the unknown for k samples in the training set. If within that neighborhood, more samples lie in class i than any other classes, the unknown sample is then assigned as class i. The confidence level for each class (i) is calculated by k_i/k, where k_i is the number of training samples that belong to class i within the neighborhood, and k is the total number of training samples in the neighborhood. A *confidence range* of each class is derived with its lower bound equal to the smallest confidence level and the upper bound equal to the largest confidence level of the class.

Fig. 2.14 illustrates the migration of a mobile agent in a distributed wireless sensor network with 3 sensor nodes. Each sensor node has generated an confidence range (CR) for each class using the algorithm described above, as shown in Fig. 2.14(a). First, the mobile agent is dispatched from node 1 and migrates to node 2 carrying CR_1 in its buffer as in Figure 2.14(b). When it arrives node 2, it combines CR_1 with CR_2 and derives a partially integrated confidence range CR_{12}, which is shown in (c). Then the mobile agent is sent out again from node 2 to node 3 carrying CR_{12}. When it arrives node 3, it will calculate another partially integrated confidence range CR_{123} using CR_{12} and CR_3, as shown in Figure 2.14(d) and (e). If the partially integrated confidence range satisfies the accuracy requirement, the migration can be stopped immediately and the mobile agent is sent back to the processing center to generate the final result. Otherwise, the mobile agent needs to continue its itinerary until the result reaching an appropriate precision.

Several field demos (SITEX02, BAE Austin, BBN Waltham) are carried out as a part of the DARPA SensIT project. In SITEX02 field demo, 70 nodes are deployed in a field of 150×200 m^2. The 70 nodes are divided into several clusters based on their geographical locations. The targets include AAV, Dragon Wagon (DW) and HMMWV. Each sensor node is equipped with three types of sensing modalities, the passive infrared (PIR) sensor to detect the target, the microphone to collect acoustic signals, and the geophone to collect seismic signals. Once the PIR



(a) Stage 1. (b) Stage 2. (c) Stage 3.



(d) Stage 4. (e) Stage 5.

Figure 2.14: The usage of mobile agent in target classification [100].

sensor detects a target, both the acoustic and seismic signals are collected. The local classification algorithm [101] analyzes the 1-second time-series data segments with a sampling rate of 512 Hz for acoustic signals and 256 Hz for seismic signals. The algorithm extracts 26 features from the data segment, runs local signal processing algorithm, and generates a local classification result. The mobile agents are then dispatched from dispatched from cluster head selected within each cluster. Each mobile agent is 512 bytes. It carries a multi-resolution integration algorithm, a partially integrated classification result, and a pre-defined itinerary. Once arrives at a node, the mobile agent integrates the local processing results on that node with its partial integration results. If the integrated result does not reach the desired accuracy requirement, the mobile agent will continue migration, otherwise it will terminate migration and return to the cluster head. The experimental results illustrated in [101] show that collaborative target classification supported by the mobile agent based paradigm can achieve very high classification accuracy (96%) than that of the single sensor processing.

Chapter 3

Cluster-based Hybrid Computing Paradigm

In the previous chapter, we discussed the computing paradigms in detail, including their modeling, performance evaluation and implementation. This chapter proposes a cluster-based hybrid computing paradigm [142] that combines the advantages of both paradigms. Simulations show the advantages of the hybrid paradigm over either the client/server-based or mobile-agent-based paradigm.

3.1 Hybrid Computing Paradigm

From the seven experiments described in Chapter 2, we can conclude that the mobile agent paradigm performs much better when the number of nodes is large. However, when the number of nodes in a network is small, the mobile agent paradigm will suffer from longer network latency because of its overhead. Therefore, the mobile agent paradigm is more suitable for large networks while the client/server paradigm has advantages in small networks. Based on this observation, we propose a cluster-based hybrid computing paradigm combining the advantages

of both computing paradigms. The idea is to divide the network into several clusters. Each cluster has a cluster head managing several nodes. We design four schemes to perform the hybrid computing.

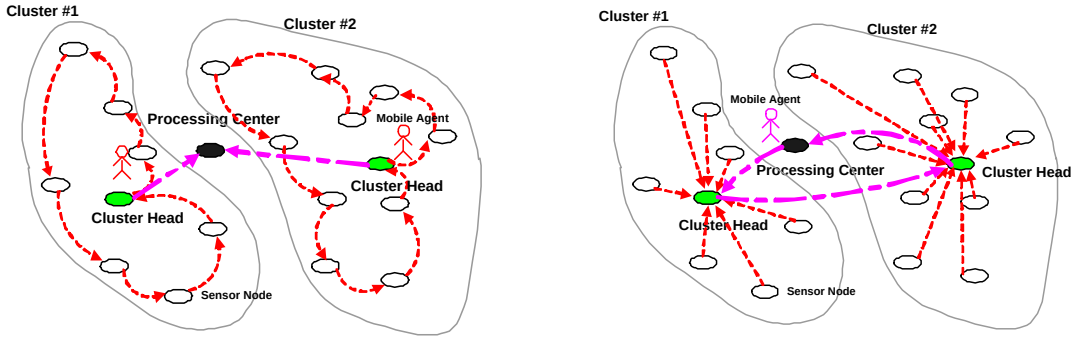
Scheme A: The mobile agent paradigm is carried out within a cluster, while the client/server model is adopted between cluster heads and the processing center, as shown in Fig. 3.1(a). The cluster head dispatches a mobile agent, which will migrate to the nodes and process data on the nodes within that cluster. After the mobile agent returns, the cluster head forms a partial result and sends it to the processing center. In this scheme, we should restrict the number of clusters so that the number of nodes within a cluster is large, but the number of cluster heads is small. Thus we can make full use of the advantages of both the client/server paradigm and the mobile agent paradigm.

Scheme B: The client/server paradigm is employed within a cluster. A mobile agent computing paradigm is formed among the cluster heads and the processing center. The nodes within a cluster send the local processing results directly to the cluster head, which will generate a partial fusion result. After a certain period, a mobile agent is sent from the processing center to integrate these results. Fig. 3.1(b) illustrates this scheme. We should choose this scheme when the number of clusters is large, and the number of nodes within a cluster is relatively small.

Scheme C: The client/server paradigm is employed within a cluster, as while as among the cluster heads and the processing center. The nodes within a cluster send the local processing results directly to the cluster head, which will generate a partial fusion result and sends it to the processing center. Fig. 3.1(c) illustrates this scheme.

Scheme D: The mobile agent paradigm is carried out within a cluster, and then the mobile agent computing paradigm is formed among the cluster heads and the processing center. Fig. 3.1(d) illustrates this scheme.

We name this new paradigm a cluster-based hybrid computing paradigm. We assume the wireless sensor network has been clustered by certain clustering algorithms, like LEACH [60].



(a) Scheme A.

(b) Scheme B.

(c) Scheme C.

(d) Scheme D.

Figure 3.1: Different schemes in the cluster-based hybrid computing paradigm.

Given the cluster setups, different schemes can be chosen accordingly. We thus develop mathematical models to quantitatively measure the performance of this paradigm.

The execution time model for scheme A of the hybrid paradigm is:

$$t_{hy-A} = t_{trans-wc} + 2(m + n_{node})o_a + t_{trans-bc} + 2n_{cluster}o_f \tag{3.1}$$

where $t_{trans-wc}$ and $t_{trans-bc}$ are the transmission time spent within one cluster and between clusters, respectively, and are obtained from simulation. n_{node} is the number of nodes within one cluster; $n_{cluster}$ is the number of clusters; m is the number of mobile agents a cluster head dispatches. We set m to 1. The other parameters have the same meaning as in Eqs. 2.3 and 2.4.

Similarly, for scheme B, the execution time model is:

$$t_{hy-B} = t_{trans-wc} + 2n_{node}o_f + t_{trans-bc} + 2(m + n_{cluster})o_a \tag{3.2}$$

For scheme C, the execution time model is:

$$t_{hy-C} = t_{trans-wc} + 2n_{node}o_f + t_{trans-bc} + 2n_{cluster}o_f \tag{3.3}$$

For scheme D, the execution time model is:

$$t_{hy-D} = t_{trans-wc} + 2(m + n_{node})o_a + t_{trans-bc} + 2(m + n_{cluster})o_a \tag{3.4}$$

The energy model for scheme A is:

$$\begin{aligned} e_{hy-A} = {} & e_{trans-wc} + 2m(n_{node} + 1)p_{processor}o_a \\ & + e_{trans-bc} + 2mn_{cluster}p_{processor}o_f \end{aligned} \tag{3.5}$$

where $e_{trans-wc}$ and $e_{trans-bc}$ are the total energy usage within one cluster and between clusters, respectively. The other parameters are the same as in Eqs. 2.8 and 2.9.

For scheme B, it is:

$$\begin{aligned} e_{hy-B} &= e_{trans-wc} + 2mn_{node}p_{processor}o_f \\ &\quad + e_{trans-bc} + 2m(n_{cluster}+1)p_{processor}o_a \end{aligned} \tag{3.6}$$

For scheme C, it is:

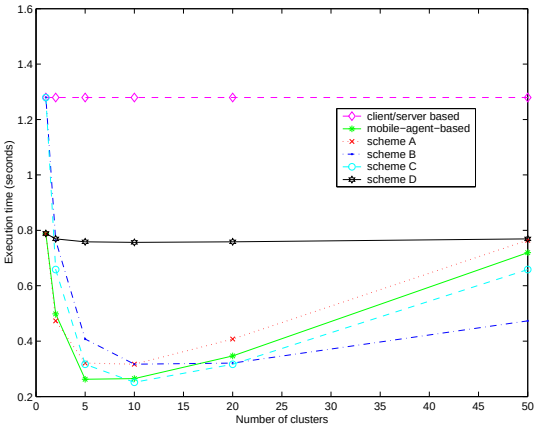
$$\begin{aligned} e_{hy-C} &= e_{trans-wc} + 2mn_{node}p_{processor}o_f \\ &\quad + e_{trans-bc} + 2mn_{cluster}p_{processor}o_f \end{aligned} \tag{3.7}$$

For scheme D, it is:

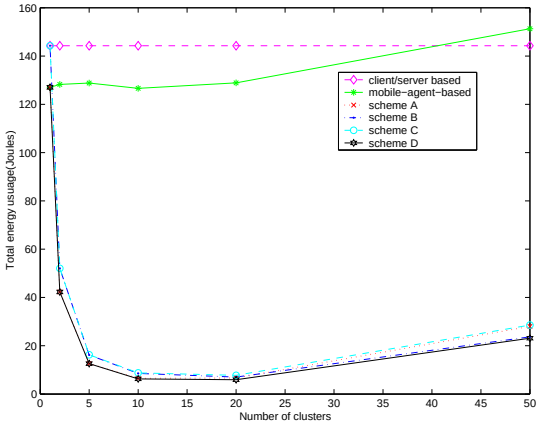
$$\begin{aligned} e_{hy-D} &= e_{trans-wc} + 2m(n_{node}+1)p_{processor}o_a \\ &\quad + e_{trans-bc} + 2m(n_{cluster}+1)p_{processor}o_a \end{aligned} \tag{3.8}$$

Based on the models derived above, we then compare the performance of the hybrid paradigm with the client/server and mobile agent paradigms. We set the number of nodes p at 100. The other parameters in the experiment are the same as in Table 2.1 and Table 2.2. The profiles of the three metrics are shown in Fig. 3.2. In each subfigure, the x-axis indicates the number of clusters for the hybrid paradigm or the number of mobile agents for the mobile agent paradigm. So when $x = 5$, the corresponding network configuration for the mobile agent paradigm is a network with 5 mobile agents, and without clustering. The client/server paradigm is not affected by the number of clusters, so we expect constant profiles as the number of clusters increases. Notice that when there is only one cluster in a hybrid computing paradigm, scheme A and D are actually the mobile agent paradigm with one mobile agent and scheme B and C reflect the client/server paradigm.

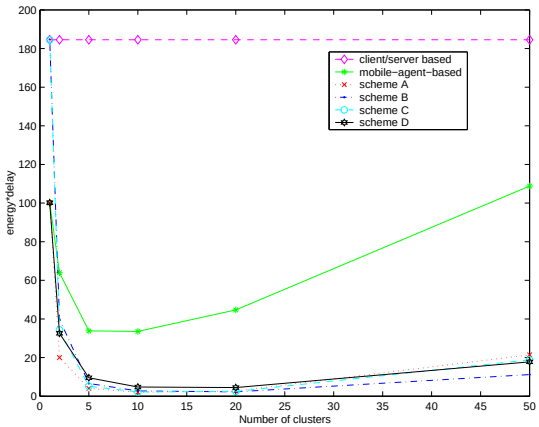
From Fig. 3.2(a) we can observe that scheme A has the lowest execution time when $n_{cluster} < 3$. This is when the network is configured to be suitable for the client/server paradigm among the cluster heads. When $9 < n_{cluster} < 20$, scheme C performs the best because now the



(a) t_{cs} vs. t_{ma}.



(b) e_{cs} vs. e_{ma}.



(c) energy*delay.

Figure 3.2: The comparison of three computing paradigms.

number of clusters and the number of nodes within a cluster both suitable for the client/server computing. After that, since now there are large number of cluster heads, scheme B is better than other computing paradigms. For the energy metric, there is always one scheme of the hybrid computing that performs better than either the client/server or the mobile agent computing. The energy*delay metric shows that among the four paradigms, scheme A performs the best before $n_{cluster} = 15$ and scheme B performs the best after $n_{cluster} = 15$.

In summary, we can see the cluster-based hybrid computing paradigm can be advantageous to both client/server and mobile agent paradigms if we choose the proper scheme according to network clustering conditions. Another observation is that the performance of the hybrid paradigm change more dramatically than either the client/server or mobile agent paradigm. This is because while for some network configurations, we can make full use of the advantages of both the client/server and mobile agent paradigm, for some other configurations, we could unfortunately inherit the disadvantages of both paradigms. An example of such an undesirable situation is when the number of clusters is large for scheme A and the number of clusters is small for scheme B.

3.2 Discussions

In this chapter, we focused on the development of a new distributed computing paradigm for CSIP in sensor networks. Based on the evaluations in Chapter 2, we presented a cluster-based hybrid computing paradigm to combine the advantages of these two paradigms. There are four schemes in this paradigm and we showed that they are advantageous to both the client/server and mobile agent paradigms under different network clustering conditions, in terms of execution time, energy usage, and energy*delay.

Problems in wireless sensor networks, with their hundreds or even thousands of nodes, and related high node-failure rates, unreliable communication links, and reduced bandwidths, can be addressed very satisfactorily using the cluster-based hybrid computing. This paradigm allows

energy efficient, high performance, scalable, and fault-tolerant solutions for collaborative signal and information processing.

Chapter 4

Mobile Agent Planning for CSIP

As we discussed in Sec. 1.4, there are five important issues for CSIP in wireless sensor networks: dense deployment of sensor nodes, asynchronous property, energy efficiency, reliability issue, and requirement of progressive accuracy. In order to facilitate collaborative processing in WSNs, it is essential to deploy an efficient computing paradigm that supports collaboration among sensors. The benefits of the mobile agent based computing in support of collaborative processing in WSNs have been thoroughly studied in Chapter 2. Chapter 3 further proposed a hybrid computing paradigm that utilized both client/server and mobile agent computing paradigms. However, the biggest hurdle in practical deployment of the mobile-agent based computing is the dynamic determination of mobile agent itinerary.

The mobile agent migration route, including the selection of nodes and the order of migration, determines the energy consumption, data fusion accuracy, mobile agent migration time, and has a significant impact on the overall performance of the sensor network. In order to combat the challenges raised by the WSN, low energy consumption, high fusion accuracy, and fewer migration hops are always the main goals in designing mobile agent migration. The derivation of a suitable mobile agent route needs to take into consideration the trade-offs between the cost on energy consumption and migration and the benefit of high accuracy from fusion, since

although visiting more sensors improves the fusion accuracy, it also increases the communication and computing overheads. Moreover, in some applications such as target detection and classification, it is critical to consider other factors that are application-specific. For example, the spatial distance between the target and a sensor node is closely related to the signal energy measured by the sensor node. The closer the target to the node, the stronger the signal energy. The mobile agent would always gain more information if migrating to the nodes with stronger signal energy.

In this chapter, we focus our discussion on determination of the mobile agent itinerary with the objectives of reducing energy consumption and improving reliability of collaborative processing in sensor networks. We refer to this problem as the *Mobile Agent Planning* (MAP) problem. We first model both static and dynamic mobile agent migrations, then propose several MAP algorithms for the mobile agent to determine a more efficient itinerary in terms of energy consumption, time consumption and network lifetime.

4.1 Related Work

The MAP-related research can be divided into two categories, the Static Mobile Agent Planning (SMAP) and the Dynamic Mobile Agent Planning (DMAP). The SMAP techniques make use of current global network information and derive an efficient path of mobile agents at a central processing center *before* dispatching the agents. In an environment as dynamic as the sensor network, in which connections between nodes can be lost or the sensor node may be malfunctioning, the SMAP techniques may not respond to the changes in real time. In the worst case, an agent can be halted forever trying to move to the target node because of an unexpectedly broken link along its itinerary. On the other hand, the DMAP techniques determine on the fly the route at each stop of the migration of a mobile agent. It is designated to provide network sensitivity with the ability to change the agent's itinerary appropriately.

Some literature [58, 90] treats the problem of SMAP as a *Traveling Salesman Problem* (TSP) or TSP variations such as the *Vehicle Routing Problem* (VRP) [52]. We make a distinction between MAP and TSP problems based on studies done in [22] that TSP deals with the optimal total routing cost with given travelers, whereas SMAP attempts to minimize the execution time to complete an information retrieval task; and that mobile agents can visit any node more than once, which is not the case in TSP.

Thus far, most research on MAP has been focused on information retrieval and data-mining. For example, Moizumi et al. [90] explored how mobile agents can efficiently schedule the sequence of nodes to visit in order to minimize the total execution time until the desired information is found. They discussed the SMAP problem from the probability point of view, where n sites are given at which a certain task might be successfully performed. The probability of success at each site is p_i. They further used dynamic programming to evaluate an approximate solution to the SMAP problem in polynomial time. Baek et al. [23] proposed an Agent Chaining approach for information retrieval. It is a dynamic planning algorithm, called k-chaining to adapt the fluctuation of network traffic.

Yang et al. [143] introduced the concept of an itinerary graph, which can not only describe the migration semantics of the mobile agent, but also reflect the changes of software and hardware environment where mobile agent resides. During its travel, the mobile agent equipped with the itinerary graph can perceive the network changes and modify its migration path autonomously.

In this section, we study the itinerary planning problem within the context of collaborative processing where energy consumption is the major concern.

The difference between the mobile agent application in data fusion and information retrieval is that a mobile agent in sensor data fusion provides *progressive accuracy*. A mobile agent always carries a partially integrated result generated by nodes it already visited. As the mobile agent migrates from node to node, the accuracy of the integrated result is constantly improved.

Therefore, the agent can return results and terminate its migration any time the integration accuracy satisfies the overall requirement. This feature, on the other hand, also saves both network bandwidth and computation time, since unnecessary node visits and agent migrations are avoided. On the other hand, a mobile agent in information retrieval does not have the progressive accuracy feature. Once it arrives at a node, it searches for the desired information. If the information is found, it returns to the home node. If not, it has to migrate to the next node.

Qi et al. [100] gave a detailed description on the use of mobile agents in target classification in wireless sensor networks. As the mobile agent migrates among the sensor nodes, the accuracy of the result increases progressively. However, the route of the mobile agent is determined *a priori*, which may deteriorate the performance of collaborative processing.

Wu et al. [138] proposed a semi-dynamic routing scheme using a two-level genetic algorithm for data fusion in sensor networks. The processing element computes an initial optimal route and then dispatches a mobile agent. If some events occur and affect the network topology, the processing element is notified which triggers the recalculation of an optimal route. The drawbacks of this algorithm is that in some situations, it is difficult to locate a mobile agent and update the optimal route it carries. Moreover, the updated optimal route may become outdated when the mobile agent receives it.

4.2 Mobile Agent Migration

In the mobile-agent-based computing, the data is processed locally. Mobile agents are dispatched from the processing element (PE) and expected to visit a subset of sensors to integrate local processing results. The mobile agent computing paradigm supports collaborative signal and information processing efficiently. It supports a wide range of applications, including distributed detection, classification, and monitoring. In such problems, although each individual sensor does not have enough information to reach a decision, a number of sensor nodes jointly may possess useful information for a sensing task. The goal of the mobile agent computing

is to migrate to a group of sensor nodes in a particular sequence, maximizing the information extracted while keeping resource usage to a minimum. Without loss of generality, we use the tracking problem as an example in this chapter. Although intuitively, the more sensors visited, the higher the accuracy achieved, it is important to select an appropriate route so that the required accuracy level can be achieved with low energy consumption, while prolonging the network lifetime.

4.2.1 Assumptions

Following assumptions are made about the sensors and sensor networks in the development of the the mobile agent planning algorithms in this chapter:

- The acoustic amplitude sensors are used and they are omnidirectional.
- The location of the sensor nodes are known, with the use of the GPS system.
- The sensors are densely deployed and each sensor node has at least two neighbors.
- The resolution of the sensors should be larger than half of the interval of between neighbor sensors.
- All sensors are calibrated.
- Sensors are deployed in a two-dimensional plane.
- All sensor nodes are synchronized.
- There is only one target in the field, and the target is moving slowly. The speed and the direction of the target do not change abruptly.
- All nodes start with the same fixed amount of energy.

4.2.2 Symbols Used for Mobile Agent Migration

In order to facilitate the modeling of mobile agent migration, we elaborate the symbols used in the context:

- t denotes the time. We consider discrete time $t \in 0, 1, \ldots, \infty$.
- $k \in 0, \ldots, n$ denotes sensor index.
- $x(t)$ the state of the event at time t. In target tracking applications, $x(t)$ is the location of the moving object in a two-dimensional plane.
- x_k: the position of sensor k.
- $z_k(t)$: the measurement of sensor k at time t.
- $Desire_{p,q}$: the desired accuracy level for a specific task.
- $Hop_{p,q}$: current number of hops $MA_{p,q}$ has.
- S_k: the kth sensor node visited by the mobile agent $MA_{p,q}$, where $k = 0, \cdots, Hop$, and where S_0 is the PE.
- $I_k(t)$: the information gain at a sensor S_k at time t.
- $Carry_{p,q}$: the total information gain the agent $MA_{p,q}$ carries so far, which is the summation of information gain at the sensor nodes the mobile agent migrates.
- E_{kj}: the energy consumption for the agent to move between nodes S_k and S_j, $k, j = 0, \cdots, n$.

4.2.3 Evaluation Metrics for Mobile Agent Migration

In order to facilitate the evaluation of different mobile agent migration algorithms, some metrics are necessary. We use three metrics in this dissertation, *energy consumption*, *network lifetime*, and *the number of hops*.

The *energy consumption* is one of the most important metrics in wireless sensor networks since sensor nodes are energy-constrained. The energy metric measures the total energy the wireless sensor network consumes to finish a processing task. The mobile agent migration process consists of the agent migrating from node to node, where each migration is called a "hop". The energy consumption for each hop is composed of three parts – mobile agent transmission energy from node k to node j (E_{kj}), agent overhead (such as agent creation, sending, and receiving) and data processing energy on node k (E_k). In this chapter, we assume E_k is identical for all sensor nodes. The total energy consumption is the total energy used for all hops to finish a task.

The *network lifetime* is defined as the time from node deployment to the time when the first node is out of service due to energy depletion. It is also an important metric since it is desirable to make the wireless sensor network function as long as possible. In order to prolong the network lifetime, we need to balance the energy consumption among sensor nodes.

The number of hops metric reflects the time spent for the mobile agent to finish a task, and thus another important metric for performance evaluation. In this section, we assume the time used to transmit and receive an agent is the same for all nodes. Once the agent arrives at a sensor node, it spends the same amount of time to process the data, and then the same amount of time to decide the next sensor node to migrate to. Thus the number of hops is the only factor that affects the total time used for mobile agent migration, and it is obviously desirable to reduce the number of hops the agent needs to finish a task.

4.2.4 Sensing Model

The measurement made by a sensor node k at time t is modeled as follows [105]

$$z_k(t) \propto \frac{E}{\| x(t) - x_k \|^2} \tag{4.1}$$

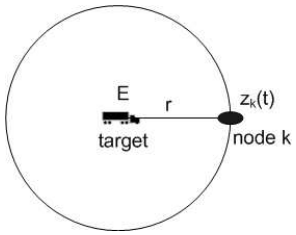


Figure 4.1: Measurement on sensor k is inverse proportional to the square of the distance to the target.

where E is the signal power emitted by the target, $x(t)$ is the target location at time t, x_k is the location of sensor node k, and $\| x(t) - x_k \|$ is the Euclidean distance between sensor node k and the target at time t. The relationship between the target and the sensor measurement is illustrated in Fig. 4.1.

4.2.5 Information Gain Model

To quantify the contribution of individual sensors to the success of its task, we introduce the concept of information gain. For acoustic sensors, we note from Eq. 4.1 that at any time instant, the measurement z_k is related to the target position $x(t)$ only through $\| x(t) - x_k \|$, the distance from the target to the sensor node k. Thus, we use this distance as a good approximation to the gain of useful information if the mobile agent migrates to the sensor node k. We model the relationship between the information gain and the target distance using a zero mean Gaussian,

$$I_k(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\|\widehat{x(t)} - x_k\|^2}{2\sigma^2}} \tag{4.2}$$

where σ is the standard deviation, a parameter that determines how fast I_k decreases when the Euclidean distance between the node and the target increases, x_k is the location of sensor node k, and $\widehat{x(t)}$ is the estimated target location calculated from the target localization algorithm explained in Sec. 4.2.7.

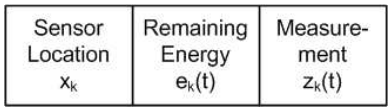


Figure 4.2: Information contained in the beacon frame.

4.2.6 Beacon Frames

We use beacons to periodically exchange information among neighbor nodes. The structure of the beacon frame is shown in Fig. 4.2, where x_k consists of the x-coordinate and y-coordinate location information of sensor k, $e_k(t)$ is the remaining energy of sensor k at time t, and $z_k(t)$ is the measurement of sensor k at time t. Sensor node k periodically broadcasts the beacon frame to its neighbors.

Beacons have the following functions:

(1) Provides location and measurement information from a neighbor node to facilitate the execution of the target localization algorithm and the mobile agent migration algorithms.

(2) The beacon frame also serves as an indication of the aliveness of the neighbor nodes. If a node does not receive the beacon frame from a particular neighbor for a certain period, then that neighbor node is deemed as out of function and the mobile agent will avoid choosing that node as the next hop.

4.2.7 Target Localization Algorithm

In order to facilitate the execution of the mobile agent migration algorithms, we need to estimate the target location at a certain time t. Based on the sensing model, if we have the measurements of the target at time t from two sensor nodes,

$$z_i(t) \propto \frac{E}{||\widehat{x(t)} - x_i||^2}$$
$$z_j(t) \propto \frac{E}{||\widehat{x(t)} - x_j||^2},$$

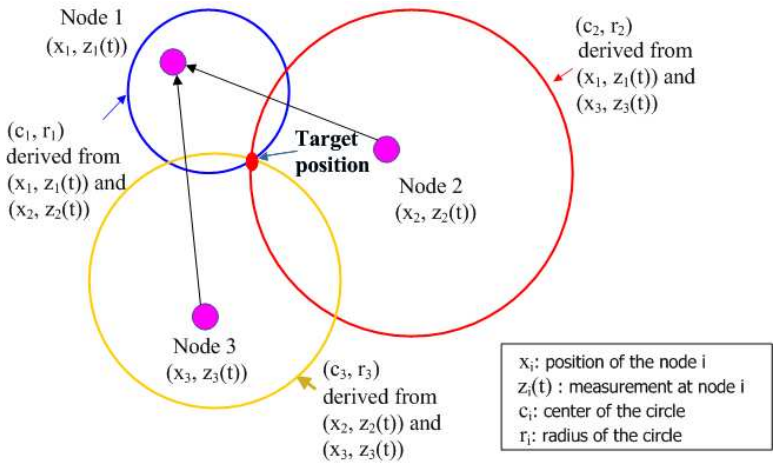


Figure 4.3: Target localization using trilateration method.

then we have

$$\frac{\| \widehat{x(t)} - x_j \|^2}{\| \widehat{x(t)} - x_i \|^2} = \frac{z_i(t)}{z_j(t)} \tag{4.3}$$

where x_i and x_j are the positions of node i and j respectively, $z_i(t)$ and $z_j(t)$ are the measurements on node i and j at time t, and $\widehat{x(t)}$ is the estimated target location at time t. Eq. 4.3 generates a circle with the center of the circle being $(\frac{x_{j1}z_j(t)-x_{i1}z_i(t)}{z_j(t)-z_i(t)}, \frac{x_{j2}z_j(t)-x_{i2}z_i(t)}{z_j(t)-z_i(t)})$, and the radius being $\frac{\sqrt{z_i(t)z_j(t)}\cdot\|x_i-x_j\|}{|z_j(t)-z_i(t)|}$, where x_{i1} and x_{j1} are the x-coordinate of x_i and x_j, and x_{i2} and x_{j2} are the y-coordinate of x_i and x_j, respectively.

In order to determine the target location, at least three measurements from different sensor nodes are needed. This target localization method is called *trilateration* [92], which is illustrated in Fig. 4.3. Although this method is more sensitive to errors in sensing compared to other target localization algorithms, such as sequential Bayesian estimation [82, 83, 152], it is computationally simpler and no previous estimation is needed. Thus, we employ the trilateration method as the target localization algorithm in this chapter.

Suppose node 1 receives beacon from node 2 and node 3, it can derive three circles indicating possible target positions based on Eq. 4.3. For example, it derives a circle (c_1, r_1) based on information from node 2 and itself, a circle (c_2, r_2) from the information of node 3 and itself. Ideally, there are two intersection points of these two circles, which indicate the possible target position, which is

$$(r_2 \cos(\arctan(\frac{c_{11}}{c_{12}}) \pm \arccos(\frac{r_2^2+c_{11}^2+c_{12}^2-r_1^2}{2r_2\sqrt{c_{11}^2+c_{12}^2}})) + c_{21},$$
$$r_2 \sin(\arctan(\frac{c_{11}}{c_{12}}) \pm \arccos(\frac{r_2^2+c_{11}^2+c_{12}^2-r_1^2}{2r_2\sqrt{c_{11}^2+c_{12}^2}})) + c_{22}),$$

where $c_{11} = \frac{x_{21}z_2(t)-x_{11}z_1(t)}{z_2(t)-z_1(t)}$ is the x-coordinate of c_1, $c_{12} = \frac{x_{22}z_2(t)-x_{12}z_1(t)}{z_2(t)-z_1(t)}$ is the y-coordinate of c_1, $c_{21} = \frac{x_{31}z_3(t)-x_{11}z_1(t)}{z_3(t)-z_1(t)}$ is the x-coordinate of c_2, and $c_{22} = \frac{x_{32}z_3(t)-x_{12}z_1(t)}{z_3(t)-z_1(t)}$ is the y-coordinate of c_2.

A third circle (c_3, r_3) derived from the information of node 2 and node 3 determines the final intersection point of these three circles.

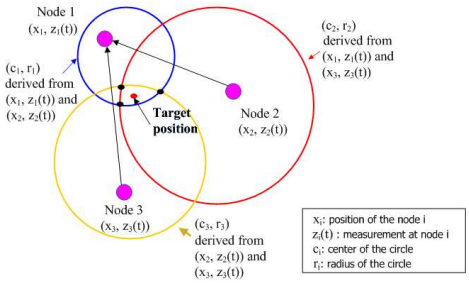
The above solution is for the ideal situation, where the three circles only have one intersection. We handle other situations of these three circles as well, as shown in Fig. 4.4.

A sensor node k receives the beacons from all its neighbors periodically. When a mobile agent arrives at node k at time t, it utilizes the first two beacon frames sent from its neighbor nodes and performs target localization algorithm described above. By using the location and measurement information from itself and from two of its neighbors, the mobile agent at sensor node k can estimate the target location at t, $\widehat{x(t)}$.

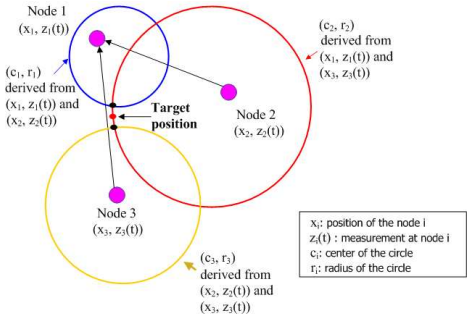
4.2.8 Procedure of Mobile Agent Migration

The procedure of the mobile agent migration can be stated in the following 3 steps:

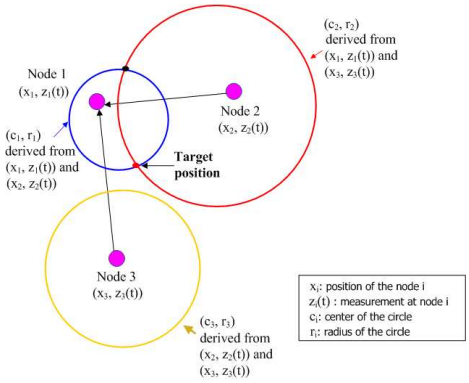
Step 1. At time $t = 0$, the sensor node that first detects the target becomes the processing center and creates a mobile agent $MA_{p,q}$. Based on the beacons it receives at $t = 0$, it derives the target location at $t = 0$ according to the target localization algorithm described in Sec. 4.2.7. It calculates the information gain I_0 using Eq. 4.2 and initializes $Carry = I_0$. It then performs



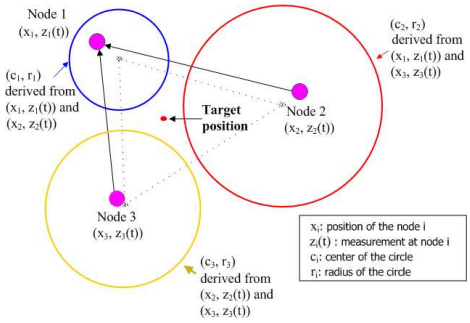
(a) Three intersection points close to each other, estimated target location is at the center of the triangle these three intersection points form.



(b) Two intersection points close to each other on an arc, estimated target location is at the center of the arc.



(c) Two intersection points close to each other and no arc, estimated target location is the intersection point that is closer to the third circle.



(d) No intersection point, estimated target location is the center of the triangle formed by the centers of the circles.

Figure 4.4: Possible conditions of trilateration.

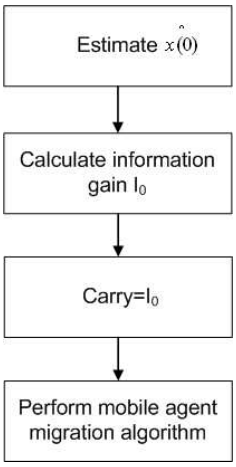


Figure 4.5: Step 1: at $t = 0$.

one of the mobile agent migration algorithms described in Sec. 4.5 to decide the next sensor node it needs to migrate to. The procedure is shown is Fig. 4.5.

Step 2. At time t, when a mobile agent arrives at a sensor node k, it first performs the target localization algorithm to estimate the target location $\widehat{x(t)}$. Then it calculates the information gain at current node k I_k and updates $Carry = Carry + I_k$. If $Carry$ is larger than a predefined information gain level $Desire$, then go to Step 3. Otherwise, it determines a neighbor node that it will migrate to by performing one of the mobile agent migration algorithms. Then go back to Step 2 and change the current time $t = t + 1$. The procedure is shown is Fig. 4.6.

Step 3. Return to the processing center.

The mobile agent carries a data buffer which is shown in Fig. 4.7, where "Previous Target Location $x(t-1)$" is the estimated target location calculated from previous migration, and $Carry_{p,q} = \sum_{k=0}^{Hop} I_k$ is the summation of the information gain from previously migrated nodes.

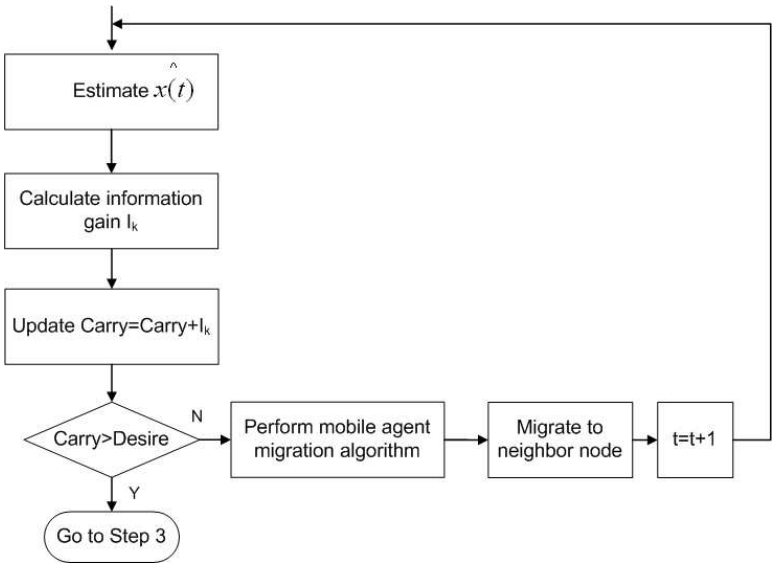


Figure 4.6: Step 2: at time t.

Previous Target Location x(t-1)	Accumulated Information Gain $Carry_{p,q}$

Figure 4.7: Data space of the mobile agent. Previous target location $x(t-1)$ is the estimated target location from previous migration, $Carry_{p,q}$ is the summation of the information gain from previously migrated nodes.

4.3 Static Mobile Agent Planning (SMAP) Modeling

The SMAP problem is in search of an order in which the sensor nodes are visited by the mobile agent, namely, a permutation $< S_1, S_2, \cdots, S_n >$. Each permutation is called a $route$, R. We employ a similar method used in [90] in our modeling. Formally, the SMAP is defined as follows:

The Static Mobile Agent Planning Problem (SMAP) - There are $n + 1$ nodes, s_k, with $0 \leq k \leq n$. Each node has a probability, $p_k \leq 1$, of being able to successfully complete the agent's task and an energy consumption E_k required for the agent to process the data at k. The energy consumption for the agent to move between nodes is given by E_{kj}. SMAP is to minimize the expected energy consumption and the expected time (in terms of the number of hops) to successfully complete the task.

Without loss of generality, we make several assumptions to simplify the SMAP model.

(1) The target is static. Therefore, $z_k(t)$ remains a constant during the mobile agent migration process.

(2) $E_{kj} = E$ for all the migration steps.

(3) $E_k = e$ for all sensor nodes.

The expected energy consumption to complete the task or visit all nodes in failure for a route $R =< S_1, S_2, \cdots, S_n >$ is:

$$E_R = E + e + p_1 E + \sum_{i=2}^{n} (\prod_{j=1}^{i-1} (1 - p_j)(E + e + p_i E)) + \prod_{j=1}^{n} (1 - p_j) E \qquad (4.4)$$

The equation can be explained as follows. The first site, S_1, is always visited and consumes E amount of energy. Upon arrival, energy e must be spent regardless of success or failure. With probability p_1, the task is successfully completed and the agent can return to node 0 with energy cost of another E. However, with probability $(1 - p_1)$, i.e., the failure rate, the agent migrates to node S_2. The expected energy consumption used by the mobile agent moving from node S_1

to S_2 is $(1-p_1)E$. Similarly, the mobile agent consumes energy E to migrate from node $i-1$ to node i, and then with probability p_i, it succeeds at node i and return to node 0 consuming energy E. So, the accumulated energy consumption at node S_i is $\prod_{j=1}^{i-1}(1-p_j)(E+e+p_iE)$. Finally, the last term arises when failure occurs at all nodes and the agent must return to the originating node 0 with an energy consumption E.

Furthermore, the expected number of hops to complete the task or visit all nodes in failure, for a route $R =< S_1, S_2, \cdots, S_n >$ is:

$$Hop_R = 1 \cdot p_1 + \sum_{i=2}^{n}(i \cdot \prod_{j=1}^{i-1}(1-p_j)p_i) + (n+1)\prod_{j=1}^{n}(1-p_j) \tag{4.5}$$

We use the following equation to model the probability of success:

$$p_k = 1 - \frac{Desire_{p,q} - \sum_{i=0}^{Hop} I_{k_i}}{I_{max}} \tag{4.6}$$

where I_{max} is the maximum information gain a sensor node can provide. Hop is the total node numbers the mobile agent has migrated through.

We then derive the following theorems.

Theorem 1 The Static Mobile Agent Planning Problem (SMAP) is NP-Complete.

Proof – We employ a similar strategy in [91] to start by showing that SMAP belongs to NP, then further show that SMAP is NP Complete.

Given a route, R, we can verify if the expected energy consumption E_R and the expected number of hops Hop_R are smaller than or equal to B by using the formulae. This verification can clearly be performed in polynomial time ($O(n^2)$ steps). Thus, SMAP belongs to NP.

Next, we show that the Hamiltonian Cycle Problem, which is NP-Complete, can be reduced to SMAP. A Hamiltonian Cycle for graph $G =< V, E >$ is a circuit that includes all the vertices V. We define SMAP with probabilities strictly between 0 and 1 and

$$E_{kj} + e_j = \begin{cases} 0 \, if \; k, j \in E \\ 1 \, if \; k, j \notin E \end{cases},$$

so that $E_k = 0$ for any vertex on an edge in E and $E_{kj} = 0$ for any edge in E. We further define

$$Hop_R = \begin{cases} Hop_R \, if \; k, j \in E \\ Hop_R + 1 \, if \; k, j \notin E \end{cases}.$$

so that the number of hops will not increase for the agent migrating on an edge in E. This can be done in polynomial time. Then the graph G has a Hamiltonian cycle if and only if the corresponding SMAP has a route with expected energy consumption and hop number of 0. To explain this, assume the graph G has a Hamiltonian cycle H. The corresponding route R in SMAP will have zero energy consumption and number of hops because all the energy consumptions are 0 and the hop number is kept at zero. On the other hand, if the route R has energy consumption of zero and the number of hops of zero, the energy consumption and the number of hops must all be 0 along this route by construction. Here we use the fact the probabilities are strictly between 0 and 1 so that the only way for the route to be 0 is for the energy consumption and the number of hops to be 0 along the route. Thus graph G has a Hamiltonian cycle since all the edges in the route R have to belong to E by construction again. Q.E.D.

Theorem 2 The optimal route for SMAP is attained in terms of energy consumption if the nodes are visited in the decreasing order of I_k, $k = 1, \cdots, n$, that is, $I_1 > \cdots > I_k > \cdots > I_n$.

Proof – We employ a similar proof method in [91]. Consider the effect of switching the order of two adjacent nodes on the route, say k and $k+1$. We call this new route as $R^{'}$. Only the kth and $k+1$st terms are affected by the switch. The terms appearing before the kth term do not contain anything involving k or $k+1$. Terms that follow the $k+1$st term, on the other hand, all contain $(1 - p_k)(1 - p_{k+1})$ in the same way. Then the difference in the expected energy

consumption is:

$$\begin{aligned} E_R - E_{R'} &= (E + e + p_k E) \prod_{j=1}^{k-1}(1 - p_j) + (E + e + p_{k+1} E) \prod_{j=1}^{k}(1 - p_j) \\ &\quad -(E + e + p_{k+1} E) \prod_{j=1}^{k-1}(1 - p_j) - (E + e + p_k E) \prod_{j=1}^{k-1}(1 - p_j)(1 - p_{k+1}) \\ &= (E + e) \prod_{j=1}^{k-1}(1 - p_j)(p_{k+1} - p_k) \end{aligned} \tag{4.7}$$

Since $p_k > p_{k+1}$, R is a better route with a smaller expected energy consumption.

This indicates that when the kth node on the route has a smaller probability than the $(k+1)$st node in making the agent complete its job, then we can decrease the expected energy consumption by switching them.

From Eq. 4.6, we find that the probability of success on a sensor node is directly related to the total information utility the mobile agent accumulates. The higher total information gain the mobile agent carries, the more likely the agent will finish the task on the current sensor node – thus the higher probability. So the optimal route for SMAP is the sequence with decreasing information gain. Q.E.D.

Similarly, we have the following theorem:

Theorem 3 The optimal route for SMAP is attained in terms of the number of hops if the nodes are visited in the decreasing order of I_k, $k = 1, \cdots, n$, that is, $I_1 > \cdots > I_k > \cdots > I_n$.

Proof – Consider the effect of switching the order of two adjacent nodes on the route, say k and $k+1$. We call this new route as $R^{'}$. Then the difference in the expected energy consumption is:

$$\begin{aligned} Hop_R - Hop_{R'} &= k \prod_{j=1}^{k-1}(1 - p_j)p_k + (k+1) \prod_{j=1}^{k}(1 - p_j)p_{k+1} \\ &\quad -k \prod_{j=1}^{k-1}(1 - p_j)p_{k+1} - (k+1) \prod_{j=1}^{k-1}(1 - p_j)(1 - p_{k+1})p_k \\ &= \prod_{j=1}^{k-1}(1 - p_j)(p_{k+1} - p_k) \end{aligned} \tag{4.8}$$

Since $p_k > p_{k+1}$, R is a better route with a smaller expected number of hops.

This also indicates that when the kth node on the route has a smaller probability than the $(k+1)$st node in making the agent complete its job, then we can decrease the expected energy consumption by switching them.

Hence the optimal route is the sequence with decreasing probabilities or information gains, so that the mobile agent can finish the task in fewer hops. Q.E.D.

Theorem 4 If $p_k = p$ for all sensor nodes, then the expected number of hops Hop_R approaches $\frac{1}{p}$ as the number of sensor nodes n increases.

Proof –

$$\begin{aligned} Hop_R &= 1 \cdot p + \lim_{n\to\infty} [\textstyle\sum_{i=2}^{n}(i \cdot \prod_{j=1}^{i-1}(1-p_j))p + (n+1)\prod_{j=1}^{n}(1-p)] \\ &= \textstyle\sum_{i=1}^{\infty} i(1-p)^{(i-1)}p + \lim_{n\to\infty}(n+1)\prod_{j=1}^{n}(1-p) \\ &= p\frac{d}{d(1-p)}(p^{-1}) + 0 \\ &= \frac{1}{p} \end{aligned} \tag{4.9}$$

This theorem shows that we need to improve the probability of success p on each sensor node in order to reduce the number of hops for the mobile agent to finish the task.

4.4 Dynamic Mobile Agent Migration Modeling

The static mobile agent planning is a centralized algorithm that determines the route of the mobile agent once for all. It may not be well suited for the distributed and dynamic natures of the wireless sensor networks. A more desirable mobile agent planning algorithm should have a distributed feature to dynamically determines the next sensor node the agent needs to migrate on the fly. Then a major issue arises naturally – how to dynamically determine which node to migrate in order to reduce energy and time consumptions, and thus prolong the lifetime of the sensor network. In this section, we propose and model the Information-driven Dynamic Mobile Agent Planning (IDMAP) problem.

The idea of the information-driven approach is proposed by Zhao et al. from PARC (Palo Alto Research Center) [75, 152]. They introduce an information-theoretic definition of the utility measure [152] and several heuristic approximations to the measure that prove to be practically useful, including covariance, Fischer information matrix, and Mahalanobis distance [38]. They then base the decision for sensor collaboration on information constraints as well as constraints on cost and resource consumption [152]. It is assumed that there is only one leader node being active at any time and its task is to select and route tracking information to the next leader. They formulate the problem of distributed tracking as a sequential Bayesian estimation problem, and an information-driven sensor querying (IDSQ) framework is developed to select a sensor which is likely to provide the greatest improvement to the estimation of target state at the lowest cost of communication and computation [82]. They further extend the information-driven method to data querying [38] and routing [83]. Wang et al. propose an entropy-based heuristic for target localization, which is computationally more efficient than mutual-information-based methods [132].

For the DMAP problem, the mobile agent needs to decide which neighbor sensor node to migrate to. But among the available neighbor nodes, not all provide useful information that improves the accuracy of the results. The task of the mobile agent planning is to select an optimal subset and to decide on an optimal order of how to finish its task in an efficient way.

In the target tracking problem, we formulate the mobile agent's selection of the next hop as follows. The mobile agent on current node, essentially, seeks the sensor that would provide the greatest amount of information gain, which is the most "informative" sensor among the neighborhood N. Besides maximizing the information gain from the neighbor nodes, we also need to reduce the energy consumption for the agent migration and prolong the lifetime of the whole wireless sensor network. The final decision of the next hop for the mobile agent is a combined consideration of gains and loses. We thus define a cost function $C_{kj}(t)$ for mobile

agent migration from node S_k to a neighbor node S_j at time moment t as:

$$C_{kj}(t) = a \cdot \frac{E_{kj}}{E_{max}} + b \cdot (1 - \frac{I_j(t)}{I_{max}}) + (1 - a - b) \cdot (1 - \frac{e_j(t)}{e_{max}}) \tag{4.10}$$

where E_{kj} is the energy consumed to transfer the mobile agent from node S_k to node S_j, $e_j(t)$ is the available energy on node j at time t, and correspondingly, E_{max} is the maximum energy consumption for the agent to migrate between two nodes, which is determined by the maximum transmission range between two nodes. I_{max} is the maximum information gain that can be provided by the sensor such that $I_j(t)$ is normalized to be between 0 and 1, e_{max} is the initial available energy on a sensor node, and we assume all nodes start with the same fixed amount of energy. a, b are the weights used to adjust the importance of these three components, and $0 \leq a, b \leq 1$.

We can see that the cost function of DMAP consists of three parts, the *energy consumption*, the *information gain*, and the *remaining energy* of a node. The energy consumption calculates the amount of energy consumed for transmitting the mobile agent and we need to reduce this part as much as possible in order to minimize the cost function. The second part is the information gain by incorporating the measurement at the neighbor node. The mobile agent should always try to migrate to a sensor node that provides more information so that more accurate estimate about the target can be obtained and the number of nodes the mobile agent need to migrate is reduced before reaching the desired accuracy for the successful completion of a certain task. Moreover, the smaller number of nodes the agent migrates, the lower the cost. The third part is related to normalized remaining energy on a neighboring node. In order to increase the lifetime of the whole sensor network, the mobile agent should always choose the next sensor node that has a higher level of available energy. The cost function takes all three factors into consideration with parameters a and b controlling the weights of these three factors according to different applications.

We formulate the energy consumption for transmitting the mobile agent from node S_k to S_j as

$$E_{kj} = P_{kj} \times T_{trans} = \alpha P_{r-thresh} \times \parallel x_k - x_j \parallel^2 \times T_{trans} \tag{4.11}$$

where P_{kj} is the transmission power of a sensor node, $\parallel x_k - x_j \parallel$ is the distance between sensor node k and node j, T_{trans} is the time spent to transmit the mobile agent, $\alpha = \frac{4\pi^2}{G_t G_r \lambda^2}$, G_t, G_r are the antenna gain factors and are chosen to be 1, λ is the signal wavelength, and $P_{r-thresh}$ is the receiver threshold. The transmission power P_{kj} can vary according to the distance between the source node and the destination node, a major energy saving approach used in the new generation Motes sensor node [42]. Since the size of a mobile agent is fixed during the whole migration process, T_{trans} remains a constant. From this equation, we see that $\parallel x_k - x_j \parallel$ is a major factor influencing E_{kj}, which indicates that the mobile agent should migrate to a neighboring node that is closer to the current node in order to reduce energy consumption.

We thus modify the the first component of the cost function by using the distance $\parallel x_k - x_j \parallel$ to indicate energy consumption over transmission, and the cost function becomes,

$$C_{kj}(t) = a \cdot \frac{\parallel x_k - x_j \parallel^2}{d_{max}^2} + b \cdot (1 - \frac{I_j(t)}{I_{max}}) + (1 - a - b) \cdot (1 - \frac{e_j(t)}{e_{max}}). \tag{4.12}$$

According to Eq. 4.2, the information gain is related to the distance between a sensor node and the target, so the final cost function we employ is:

$$C_{kj}(t) = a \cdot \frac{\parallel x_k - x_j \parallel^2}{d_{max}^2} + b \cdot \frac{\parallel \widehat{x(t)} - x_j \parallel^2}{dt_{max}^2} + (1 - a - b) \cdot (1 - \frac{e_j(t)}{e_{max}}), \tag{4.13}$$

where $\parallel \widehat{x(t)} - x_j \parallel$ is the distance from node j to the estimated target location at time t, and dt_{max} is the maximum possible distance from a node to the target.

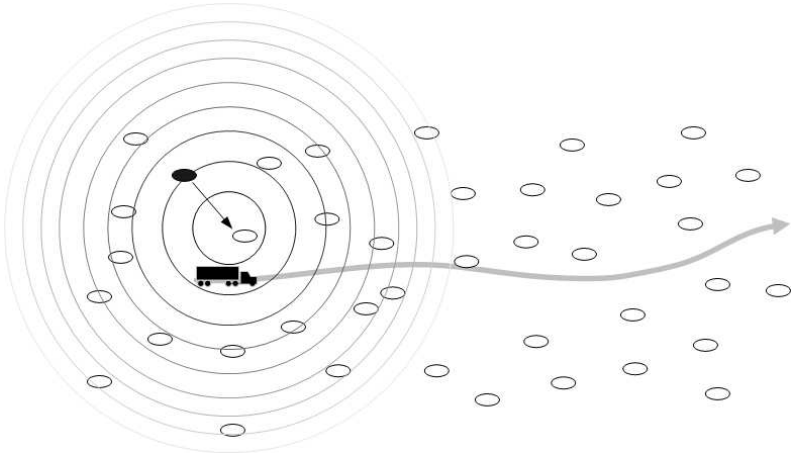


Figure 4.8: Mobile agent migration by optimizing an objective cost function.

Finally, the decision is

$$j^* = \arg\min_{j \in N_k} a \cdot \frac{\| x_k - x_j \|^2}{d_{max}^2} + b \cdot \frac{\| \widehat{x(t)} - x_j \|^2}{dt_{max}^2} + (1 - a - b) \cdot (1 - \frac{e_j(t)}{e_{max}}), \quad (4.14)$$

where N_k is the set of neighbor nodes of node k, and node S_j is the neighboring node of S_k, that is, a node which consumes less energy, senses a larger magnitude of target signal, while having more available energy.

The cost function is a combined effect of information gain, energy consumption and remaining energy. Fig. 4.8 shows a snapshot of a sensor node evaluating the cost function locally at a certain time. The cost function's iso-contours are shown as the set of concentric circles, with a smaller radius representing smaller cost function value. Here we assume the remaining energy on all sensor nodes are the same. Note that the node with minimum amount of cost value is located between the current node (indicated by the solid circle) and the target, which indicates the tradeoff between information gain and communication cost. Since the current node and the

target is constantly changing, Fig. 4.8 is only a snapshot of the cost function at a particular time, and the cost function will dynamically change.

The total cost for a mobile agent $MA_{p,q}$ to migrate is:

$$\begin{aligned} &Total_{p,q} = \textstyle\sum_{k=0}^{Hop-1} C_{i_k i_{k+1}} \\ &s.\,t.\ \ Carry_{p,q} = \textstyle\sum_{k=0}^{Hop} I_k > Desire_{p,q}, \end{aligned} \tag{4.15}$$

where k is the set of nodes the mobile agent migrates and Hop is the actual number of hops the mobile agent migrates. For the collaborative processing, the mobile agent can achieve progressive accuracy as it migrates and accumulates more information. Once it accumulates enough information, the estimation to the event is accurate enough and the agent will terminate migration and return to the processing center. So, the number of hops the mobile agent migrates is a random variable from 1 to the total number of nodes in one cluster and is determined by how fast the agent can accumulate desirable amount of information. It is an important factor affecting the total energy usage, network lifetime and execution time. The accuracy is determined by the information gain accumulated by the mobile agent. The higher the information gain, the higher accuracy the result will be. It is reasonable to choose a neighbor with higher information gain, so that mobile agent can have fewer hops to reach the desired accuracy level.

4.5 Mobile Agent Planning Algorithms

Let PE_p represent a certain processing element with an identification p that is in charge of the surveillance of a certain area. Let $\{MA_{p,1}, \cdots, MA_{p,m}\}$ represent a group of m mobile agents dispatched by PE_p. Without loss of generality, we assume that each $MA_{p,q}$ visits the same number of sensor nodes, denoted by n. We would like to choose an optimal itinerary that consumes the least amount of resources (energy and time) in order to finish the collaborative processing task. In order to do so, some information is needed for effective mobile agent

planning, including either the *global information* or the *local information*. The SMAP problem needs the global information in order to generate the optimal itinerary. The global information is a complete picture of the whole network. Such information is usually difficult to obtain in WSNs, since each sensor node has a limited transmission range. Without collaborations with other sensor nodes, it is very difficult for one sensor node to get the *global information*. The *local information*, on the other hand, only contains information of those nodes that are within the transmission range of one sensor node; it is thus a partial picture of the whole network. The objective of DMAP is to achieve *global optimization* while utilizing *local information*.

A sensor node can obtain its *local information* by periodically sending an update message to neighboring nodes or using the piggyback technique, that is, when a host needs to send a packet to its neighboring host, it attaches its current information, such as current remaining energy, current measurement, etc., along with the packet.

4.5.1 Information-driven Static Mobile Agent Planning (ISMAP)

We first propose an Information-driven Static Mobile Agent Planning (ISMAP) algorithm, which utilizes the *global information*. Before PE_p sends out a mobile agent, it collects the *global information* of the current network by communicating with all other nodes within its monitoring territory, and derives a mobile agent itinerary based on Theorem 2 and Theorem 3. Fig. 4.9 describes the procedure of the ISMAP.

Although a good itinerary can be achieved, the static itinerary planning has some drawbacks that prevent it from being practically applied. First, since this approach utilizes the *global information*, the transmission range of the sensor nodes has to be large enough in order for the message to be delivered from the node to the PE within one hop. As a result, more energy would be consumed on transmission. This approach is only suitable for a sensor network cluster of small area. Second, since it is a static algorithm, the global information collected at the PE

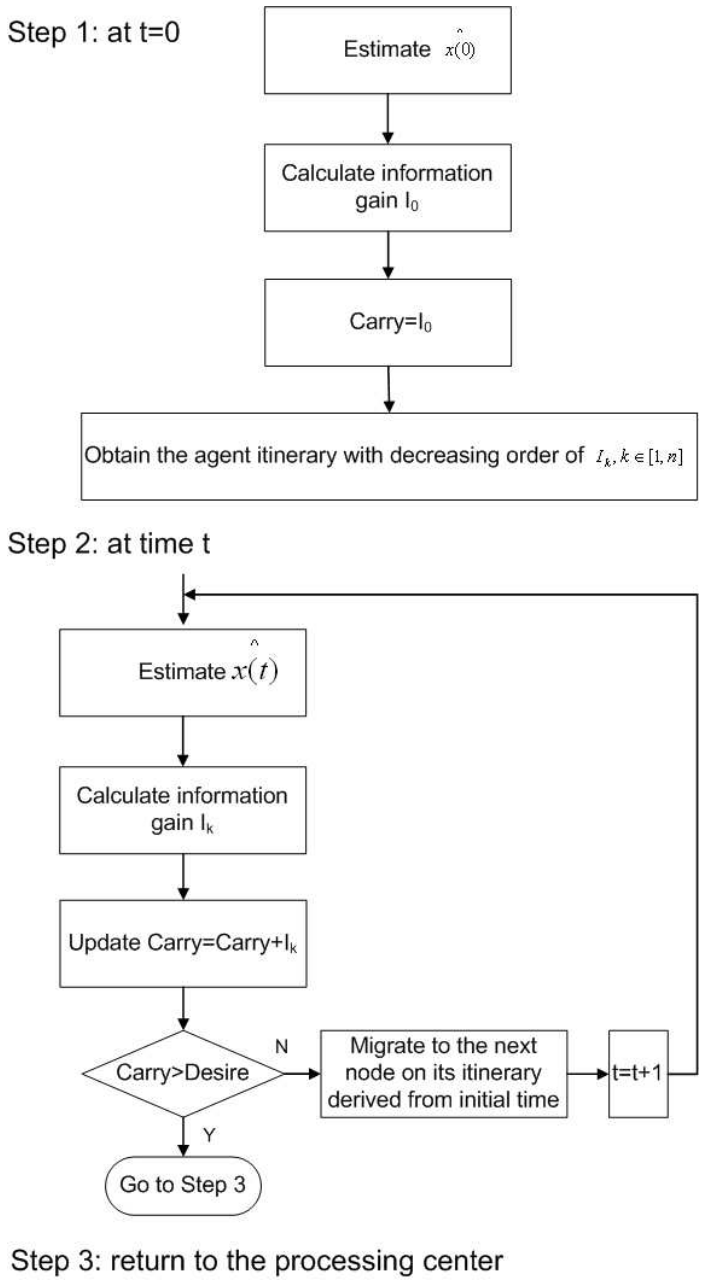


Figure 4.9: Algorithm 1: Information-driven Static Mobile Agent Planning (ISMAP)

might not reflect changes occurred during agent migration. In a dynamic network with high node failure rates or a moving target event, the static itinerary might not end up as optimal.

4.5.2 Information-driven Dynamic Mobile Agent Planning (IDMAP)

In order to combat the drawbacks of static mobile agent planning, we describe a dynamic mobile agent planning algorithm IDMAP that determines the next hop of mobile agent on the fly, according to the current network situation. Once the mobile agent arrives at a node, it finds, among its unvisited and alive neighbors, the one with the smallest cost as calculated from Eq. 4.12, such that a near-optimal itinerary can be determined. If all the neighbors have been visited, it will migrate back to the sensor node where the agent was dispatched. The agent will stop migration when the collaborative processing result it carries has reached the desired accuracy level, $Desire_{p,q}$, or when it is back to the processing center and there are no unvisited neighbors.

Since each node periodically receives the beacon information from its neighbors, DMAP algorithm always utilizes the updated *local information* in determining the next migration stop. As only *local information* is needed, sensor nodes do not necessarily need a large transmission range, and thus can reduce the energy consumption on transmission. Moreover, unlike the static itinerary planning algorithm, which would fail the collaborative processing task if some link along the mobile agent itinerary is broken, this dynamic algorithm can be informed about the broken link by not receiving the beacons from some neighbor and thus bypass that broken link. Therefore, the dynamic algorithm has a better fault-tolerance performance. The procedure of DMAP algorithm is shown in Fig. 4.10.

4.5.3 Predictive Information-driven Dynamic Mobile Agent Planning (P-IDMAP)

In collaborative processing applications, the movement pattern of the target is an important factor that affects the final fusion results. The dynamic algorithm does not consider the moving

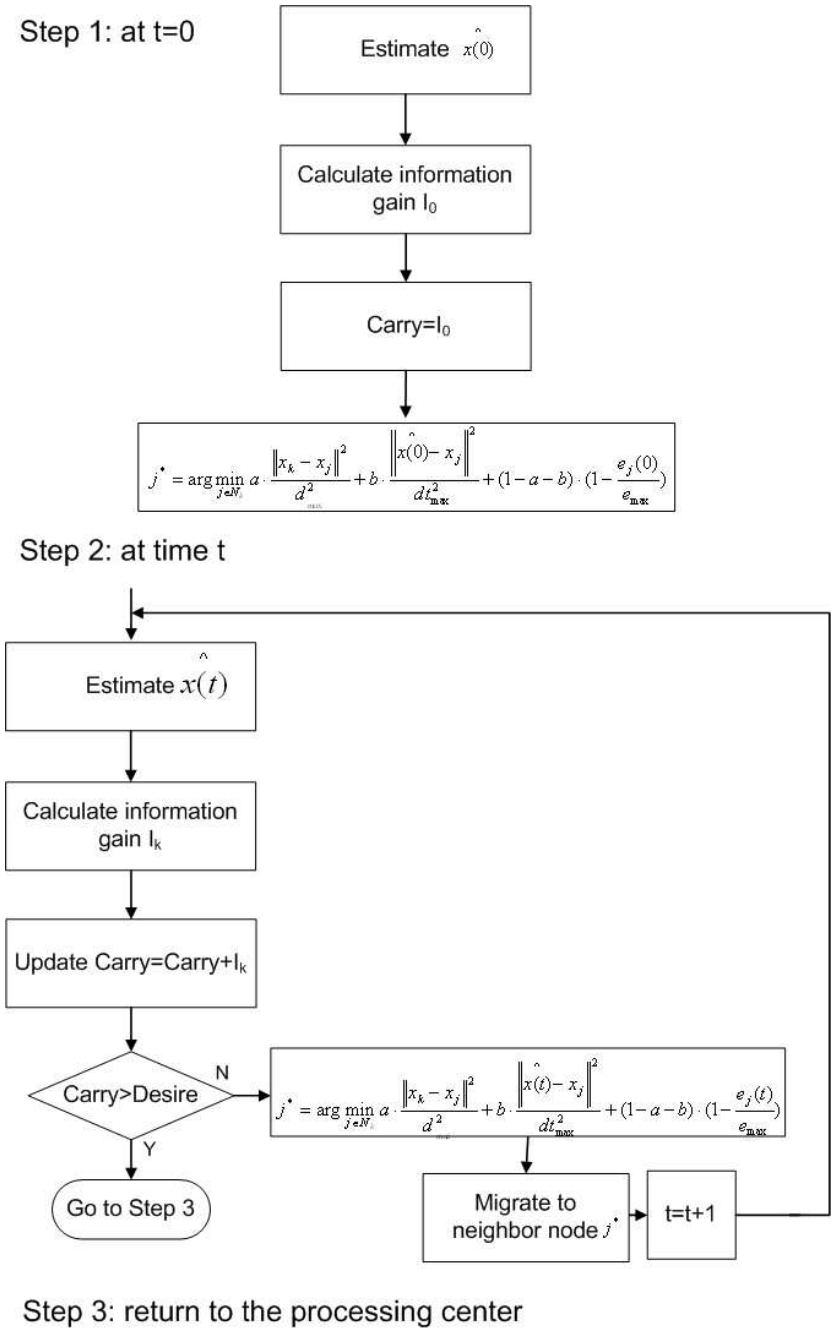


Figure 4.10: Algorithm 2: Information-driven Dynamic Mobile Agent Planning (IDMAP)

(a) Mobile agent migration at time t.

(b) Mobile agent migration at time t+1.

Figure 4.11: Effect of not using the target movement information.

direction of the target. In some situations, it may not be able to choose the best neighbor node as the next hop if the target movement information is not considered. Fig. 4.11 shows one such case. At a particular time t, as shown in Fig. 4.11(a), the mobile agent is on node A, which has two equal-distance neighboring nodes B and C to migrate, that is, $d_{AB} = d_{AC}$. At time t, when the mobile agent needs to make a decision for the next hop on node A, node B is closer to the target than node C, thus we can assume the information gain on node B is larger than that on node C, and the cost to migrate to node B is equal to that to node C. According to DMAP algorithm, the mobile agent will migrate to node B. But when the mobile agent arrives at node B, say at time $t + 1$, the target has already moved away, say, to a position that is closer to node C, as Figure 4.11(b) shows. Therefore, at time $t + 1$, the measurement and information gain on node B is less than those on node C, and the mobile agent will collect less information on node B than on node C. A better algorithm is that the agent can *predict* the direction of the target movement and migrate to the node that is *more likely* to provide more information when the mobile agent arrives at that node.

In essence, the mobile agent migration should follow the movement of the target. We call this new algorithm the *Predictive Information-driven Dynamic Mobile Agent Planning (P-IDMAP)*.

As in most cases in real situations, we assume the target dynamics are small, which means the target does not change the direction and speed of movement abruptly. That is, within very short time interval, the direction and the speed of the target can be deemed as constant. So, the target changes between the time interval $t-1$ to t, and t to $t+1$ are the same:

$$x(t+1) - x(t) = x(t) - x(t-1) \tag{4.16}$$

Eq. 4.16 requires the target location at previous moment $x(t-1)$ and target location at the current moment $x(t)$ in order to predict the target location at future moment $x(t+1)$. At time t, the mobile agent on the current node k performs the target localization algorithm to calculate $\widehat{x(t)}$, the estimated target location at time t. In addition, the mobile agent carries $x(\widehat{t-1})$, the estimated target location at $t-1$ calculated from the previously migrated node. Then the mobile agent predicts the target location at future time $t+1$ by using:

$$x(\widehat{t+1}) = 2\widehat{x(t)} - x(\widehat{t-1}) \tag{4.17}$$

It then uses an updated cost function to evaluate the cost to its neighbor nodes and chooses a neighbor node with the minimal cost function value to migrate to. The new cost function for the P-IDMAP is

$$C_{kj}(t) = a \cdot \frac{\parallel x_k - x_j \parallel^2}{d_{max}^2} + b \cdot \frac{\parallel x(\widehat{t+1}) - x_j \parallel^2}{dt_{max}^2} + (1-a-b) \cdot (1 - \frac{e_j(t)}{e_{max}}) \tag{4.18}$$

where $\parallel x(\widehat{t+1}) - x_j \parallel$ is the distance from node j to the predicted target location at time $t+1$.

Correspondingly, the decision of the next node to migrate to is

$$j^* = \arg\min_{j \in N_k} a \cdot \frac{\parallel x_k - x_j \parallel^2}{d_{max}^2} + b \cdot \frac{\parallel x(\widehat{t+1}) - x_j \parallel^2}{dt_{max}^2} + (1-a-b) \cdot (1 - \frac{e_j(t)}{e_{max}}) \tag{4.19}$$

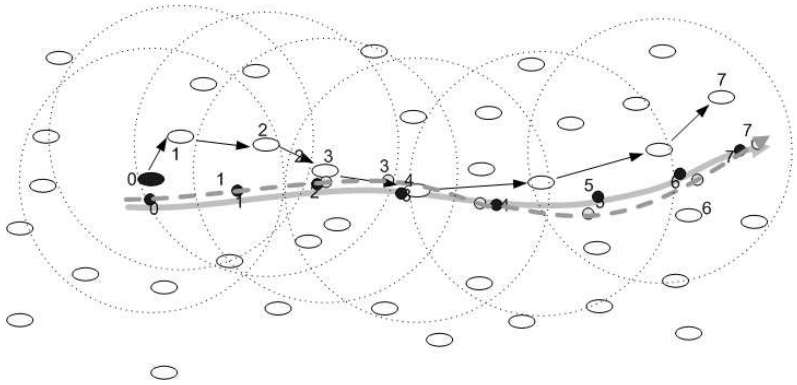


Figure 4.12: Predictive Mobile Agent Migration. The black dots represent the estimated target location, the grey dots represent predicted target position, the dashed track represents the predicted target movement.

where N_k is the set of neighbor nodes of node k.

At time $t = 0$, since the previous target location $x(\widehat{t-1})$ is unavailable, the mobile agent performing P-IDMAP algorithm utilizes the same cost function as the IDMAP algorithm and has the same procedure as IDMAP algorithm.

The mobile agent can migrate to a sensor node that is close to the predicted position of the target, so that it can have a larger measurement and information gain when it arrives at the node, which is the main idea of the predictive dynamic mobile agent planning algorithm. Of course, this algorithm is also information-driven and considers the combination of the costs of migration and information gains. The mobile agent migration procedure using P-IDMAP algorithm is shown in Fig. 4.12. The solid line represents the real target movement and the labels $0, \ldots, 7$ represent the target position at discrete time moments. The black dots represent the estimated target location after performing the target localization algorithm on the current sensor node. The dashed line represents the predicted target movement, with the gray dots representing predicted target position at discrete moments $0, \ldots, 7$ and are calculated using the current and the previous black dots.

The procedure of P-IDMAP algorithm is shown in Fig. 4.13.

4.6 Simulation and Algorithm Evaluation

Fig. 4.14 shows the mobile agent itineraries using different algorithms from Sec. 4.5. We use target tracking as an application example for the collaborative signal and information processing. Here, a vehicle moves through the sensor field from left to right. Circles represent sensor nodes, and the solid circle represents processing center (PE), where the mobile agent is dispatched. The numbers $0, \ldots, 7$ under the target indicate the target locations at time index $0, \ldots, 7$. Large dash circles represent the range of radio communication from each node. A sensor node considers all the sensor nodes within its communication range when deciding the next hop of migration. The numbers $0, \ldots, 7$ under the sensor nodes represent the kth sensor node on the mobile agent itinerary. Moreover, the mobile agent at time k is at node k.

The migration itinerary using the ISMAP algorithm is shown in Fig. 4.14(a). Since the itinerary is decided at time 0, the sensor nodes on the itinerary are the nodes close to the target at time 0. Thus, ISMAP is more suitable when the target is static or moves very slowly. The migration itineraries using IDMAP and P-IDMAP are shown in Fig. 4.14(b) and Fig. 4.14(c) respectively. We can see the mobile agent can follow the track using both algorithms, so the mobile agent can obtain more information gain than the ISMAP algorithm. However, the average distance between the target to the current sensor node is larger using IDMAP than using P-IDMAP, indicating that the mobile agent using the IDMAP algorithm cannot accumulate as much information as the one using P-IDMAP.

We have developed a simulator in JAVA to evaluate the performance of these three algorithms. We set up a basic network of 20 m $\times$ 20 m, where nodes are *grid* deployed as an array with the same number of nodes in each row and column. Table 4.1 shows the parameters for the basic network setup. The initial energy on all the nodes is 36 Joules. The transmission range of each sensor node is 10 m for the dynamic algorithms and 30 m for the static algorithm. The

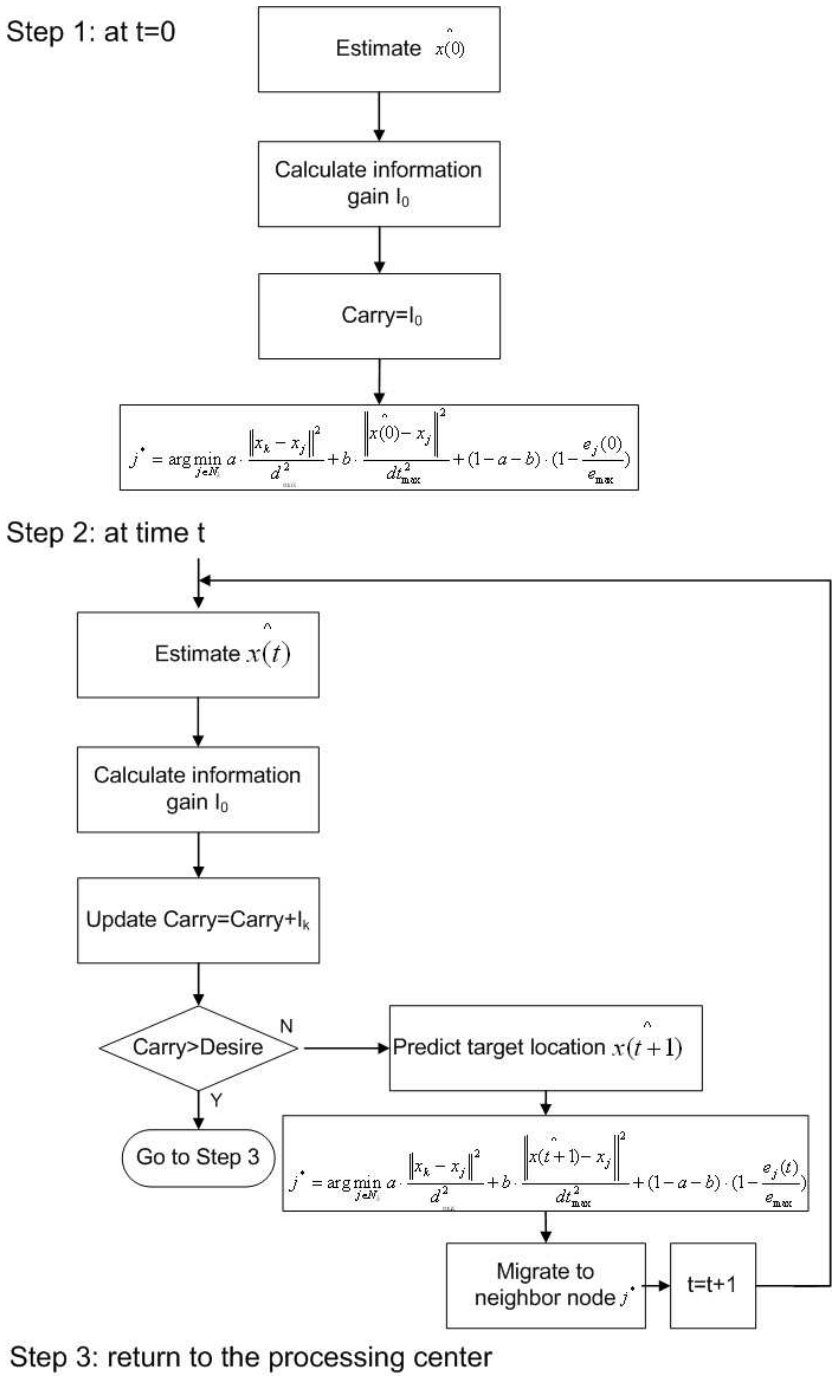
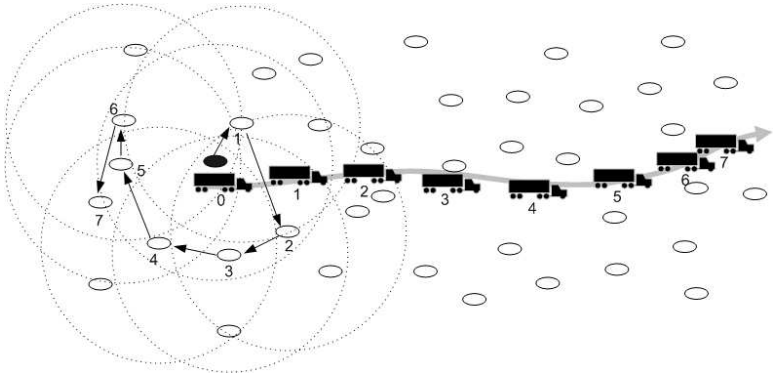
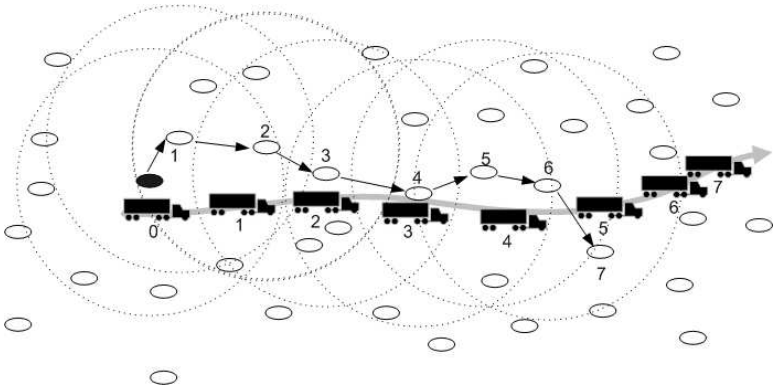


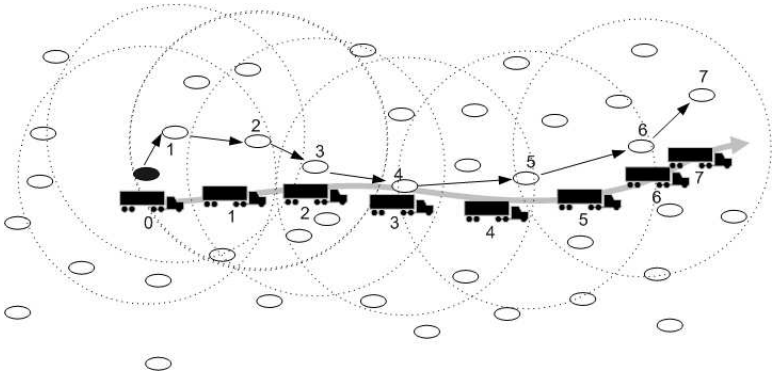
Figure 4.13: Algorithm 3: Predictive Information-driven Dynamic Mobile Agent Planning (P-IDMAP)



(a) Algorithm 1: ISMAP.



(b) Algorithm 2: IDMAP.



(c) Algorithm 3: P-IDMAP.

Figure 4.14: The mobile agent migration using different algorithms.

Table 4.1: Related parameter setup for the basic network.

Network area	20 m by 20 m
Number of nodes	500
Node placement	Grid
Sensing range	10 m
Target speed	20 m/s
Beacon interval	0.1 s
Simulation time	100 s
Desired information gain	18 units
Mobile agent size	800 bits

static algorithm needs longer transmission range in order to collect global information before dispatching the mobile agent. All nodes within the transmission range are neighbors. Sensor nodes broadcast a beacon frame every 0.1 s, which carries the current information of remaining energy, target signal, etc., on that node. The target starts at the center of the surveillance area, and moves toward the corner of the area along a straight line with a speed of v m/s. Once it moves out of the area, it reverses its moving direction and continues moving. The sensing range of a node is 10 m. Within this range, the sensor node can sense the event of the target movement and have a measurement determined by Eq. 4.1. Outside of this range, the sensor node cannot sense the event and the measurement is 0.

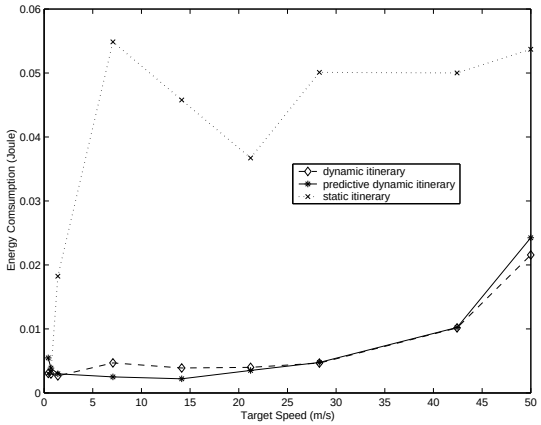
We design two experiments to evaluate the effects of different parameters on the algorithm's performance. These parameters include the target speed and the number of nodes. We use three metrics: *energy consumption*, *network lifetime*, and *number of hops* for the evaluation. In each experiment, we change one of the parameters of the basic network and keep all the others unchanged.

4.6.1 The Effect of the Target Speed (v)

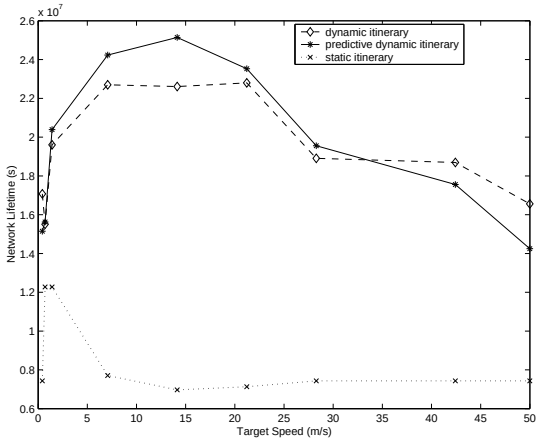
In this experiment, we keep the number of sensor nodes at 484 and change the target speed v from 0.4 m/s to 50 m/s. The results are shown in Figure 4.15. From the total energy consumption as shown in Figure 4.15(a), it can be observed that the IDMAP and the P-IDMAP algorithms perform better than the static itinerary algorithm in most cases. Moreover, the P-IDMAP consumes less energy than the IDMAP when the target speed is between 1.4 m/s and 42.5 m/s, a speed range that is typical for most target types. The reason for this is that it is more difficult to predict the movement of the target when the speed is too slow or too fast. Moreover, when the target speed is less than 0.1 m/s, the static itinerary algorithm performs a little better than the other two, as in such cases, the sensor network is less dynamic and the static itinerary algorithm can generate an optimal itinerary. Figure 4.15(b) and Figure 4.15(c) present similar patterns in terms of network lifetime and the number of hops, that the P-IDMAP algorithm performs better when the target moves within a certain range of speed.

4.6.2 The Effect of the Number of Nodes

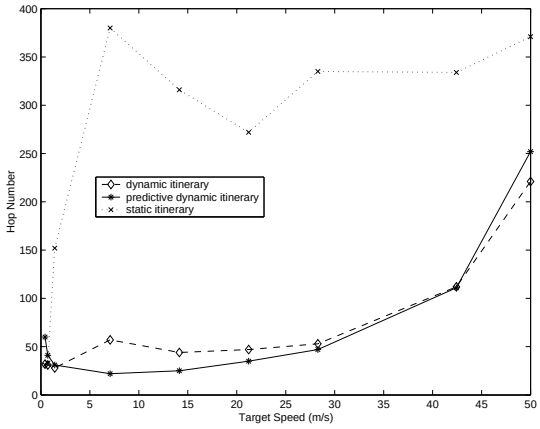
In this experiment, we keep the target speed at 10 m/s and change the number of nodes from 49 to 1156. Since the area is fixed, the number of nodes is directly related to the node density – the larger the sensor nodes, the higher the node density. The results are shown in Figure 4.16. We can see that in most cases, the P-IDMAP performs the best in terms of energy consumption, network lifetime, and the number of hops. When the number of nodes is less than 50, however, the static itinerary performs better than the other two with a smaller number of hops and longer network lifetime, while the energy consumption is almost the same among the three planning algorithms. The reason is that when the network density is too sparse, which is the case when the node number is less than 50, for the IDMAP and the P-IDMAP approaches there are too few neighbors for the mobile agent to migrate. Once all the neighbors of a node have been visited by the agent, it has to migrate back to the node already visited, which consumes energy



(a) Energy consumption.

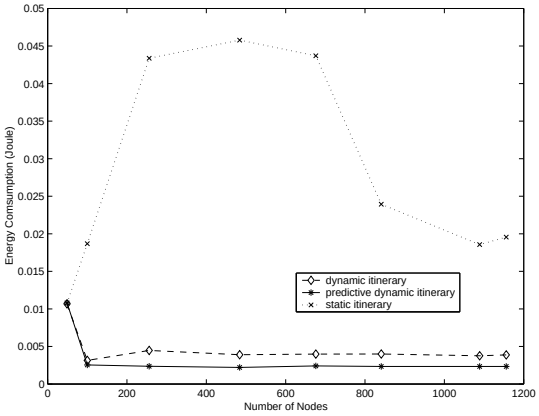


(b) Network lifetime.

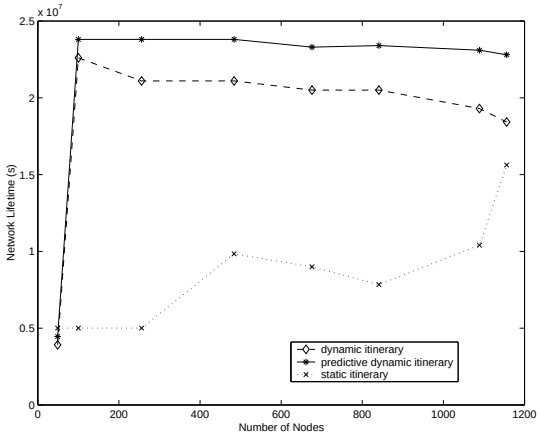


(c) The number of hops.

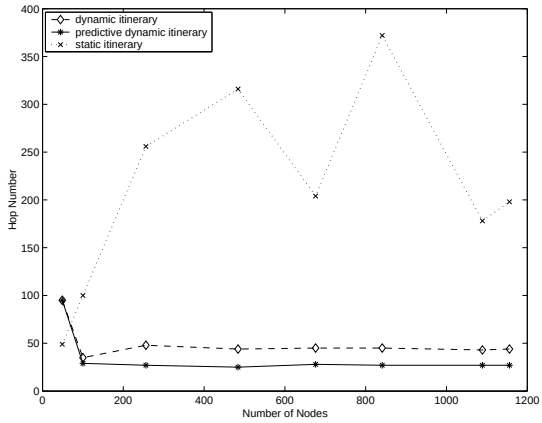
Figure 4.15: The effect of target speed.



(a) Energy consumption.



(b) Network lifetime.



(c) The number of hops.

Figure 4.16: The effect of the number of nodes.

and wastes extra hops that do not contribute to the progressive increase of total information gain. In another word, the *local information* it utilizes to decide the next hop is not sufficient. The static itinerary, on the other hand, performs better because it can still utilize the *global information*, and the transmission range of a node is large enough to dispatch the agent to any other nodes in the area. Hence the mobile agent can take lesser number of hops to reach the desired target signal level. When the node density is high enough, the mobile agent has more neighbor nodes to migrate from the current node, and the IDMAP and the P-IDMAP approaches begin to perform better than the static itinerary approach. We also observe from the performance curves that the IDMAP and the P-IDMAP approaches are not greatly affected by node density, as compared to the static itinerary approach.

4.7 Conclusion

This chapter focused on the discussion of three mobile agent itinerary planning algorithms in wireless sensor networks. In specific, it evaluated the performance of the ISMAP, IDMAP, and P-IDMAP planning algorithms. We first presented the mathematical models for both the static mobile agent planning and the dynamic mobile agent planning as optimization problems. We then designed several experiments to investigate the effect of different parameters on the performance of the algorithms. We showed, through simulation, that the ISMAP algorithm is more suitable for static or low-speed target, or sparse networks, while the P-IDMAP algorithm is suitable for a wide range of conditions. The P-IDMAP algorithm has overall advantages over other algorithms in terms of energy consumption, network lifetime, and the number of hops. It provides an energy-efficient, near-optimal, and fault-tolerant itinerary solution for collaborative processing in wireless sensor networks.

Chapter 5

Decentralized Reactive Clustering (DRC) in Collaborative Processing

In Chapter 3, a cluster-based hybrid computing paradigm was proposed, which combines the advantages of both the client/server and the mobile agent paradigms, by dividing the wireless sensor network into several clusters. Experiments show that the cluster-based hybrid computing paradigm can always be advantageous to both the client/server and the mobile agent paradigms if a proper scheme can be chosen according to the network clustering conditions. The key issue in this hybrid computing paradigm is how to cluster the network and which computing paradigm to choose accordingly. Moreover, in Chapter 4, the sensor nodes periodically receive the beacons from neighbor nodes, which contains information of the neighbor nodes, including remaining energy, current information gain, current transmission power, etc. The beacon serves as an indication of the connection of the wireless link and is used by the mobile agent to determine the next sensor node it needs to migrate to. Thus, an underlying protocol is needed in order to facilitate the migration of the mobile agent. In this chapter, a new clustering protocol called DRC is developed [141] to facilitate collaborative processing in wireless sensor

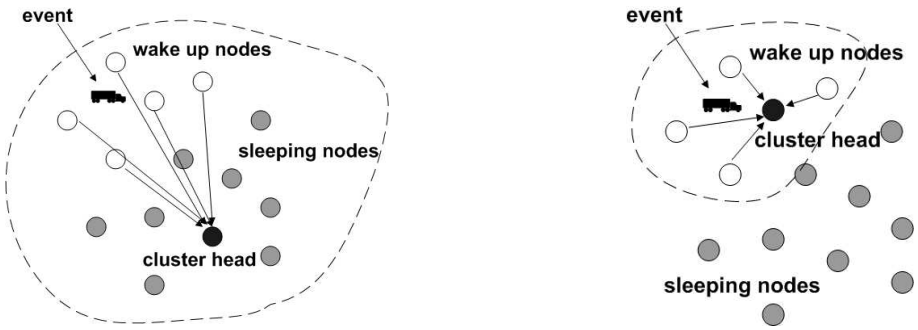
networks, which is designed specifically for the event-driven CSIP applications. This protocol is also used for periodically sending beacons to guide the mobile agent migration.

5.1 Motivation

Collaborative processing applications include collaborative detection, classification and tracking of targets, active sensor querying, etc. If the events do not occur frequently, then from an energy conservation point of view, the wireless sensor network should usually stay at the monitoring state, that is, the sensor node should mostly stay in the *sleep* mode. When an event is detected, the node wakes up its processor to collect and analyze the data. The radio, which is normally turned off, is only waked up if the processor decides that the information needs to be forwarded to other nodes. Since the sensors have limited sensing capability and computing resources, collaboration among sensor nodes is important in order to compensate for each others' limitations. Because the sensor network is usually deployed in a very large scale, collaboration has to be done locally. In addition, data correlation is the strongest among nodes that are close to each other. Therefore, the use of a clustering infrastructure would improve the effectiveness of collaboration, where sensors within a certain neighborhood form a cluster and a cluster head is automatically selected. Communications only occur between sensors within the same cluster or between cluster heads.

Clustering techniques can aid in reducing energy consumption [61] and combating large scalability. The essential operation in clustering is to select a set of sensor nodes as cluster heads and cluster the rest of the nodes with these cluster heads. Cluster heads are responsible for the nodes within their clusters and communicate with each other and with the processing center on behalf of their clusters.

To the best of the author's knowledge, previous clustering protocols are all proactive, which means that the clusters are formed in advance. Fig. 5.1(a) illustrates an example of proactive clustering. In order to save energy consumption, all sensor nodes are put into the sleep mode,



(a) Proactive clustering.

(b) Reactive clustering.

Figure 5.1: Comparison of proactive and reactive clusterings.

and only the nodes that sense the event wake up and participate the collaborative processing. As can be seen from Fig. 5.1(a), the proactive clustering may cause energy inefficiency and may not be suitable for collaborative processing because a node may have to communicate with a cluster head far away from the phenomena. So we present a decentralized reactive clustering (DRC). It has two unique characteristics.

First, DRC is a *reactive* clustering algorithm, which means that the clustering procedure is invoked only by events. Initially, the sensor nodes all enter the *sleep* mode in order to save energy. In the case of an event, the sensing units on the nodes within a certain range of the event detect it and wake up the transceiver and processing units. The awakened nodes then execute the DRC protocol and form clusters locally. Each node can be aware of other nodes in its communication range by listening to the communication channel, since the wireless channel is a broadcast medium. Fig. 5.1(b) shows an example of reactive clustering. We can see that the transmission distances to reach the cluster head are much less than those using proactive clustering.

Second, DRC employs power-control techniques to lower the energy consumption. From [112], we know the receiving power is a constant, while the transmission power can be adjusted. The higher the transmission power, the longer the transmission range, and the more energy consumed. To communicate between two nodes, the most energy-efficient situation is when the transmission power is high enough to just reach the receiving node, so that the transmission power will not be wasted. DRC achieves this goal by taking advantages of the *power control techniques* [128]. That is, the radio transmission power can be adjusted so as to control the communication range and the topology of the network accordingly. There exist several radios that can transmit with several discrete power levels, including SINCGARS [12], Rockwell WINS [17], etc. DRC can adjust the transmission power and form clusters according to the information exchanged between neighboring nodes.

Many clustering algorithms have been proposed in the past [20, 24, 36, 80, 109], but to our knowledge, none of these algorithms aim at energy efficiency in the sensor network. Most of these algorithms are heuristic and aim at generating the minimum number of clusters such that a node in any cluster is at the most d hops away from the cluster head. Recently, many clustering protocols have been proposed for sensor networks and ad hoc networks. We discuss some of these protocols in more detail.

In [37], the authors proposed a clustering algorithm that aims at maximizing the lifetime of the network by determining optimal cluster size and optimal assignment of nodes to cluster heads. They assume that the number of cluster heads and the location of the cluster heads are known *a priori*, which is not possible in all scenarios. Moreover the algorithm requires each node to know the complete topology of the network, which is generally not possible in the context of large sensor networks.

The Weighted Clustering Algorithm (WCA) [34] elects a node as a cluster head based on the number of neighbors, transmission power, battery life, and mobility rate of the node. The

algorithm also restricts the number of nodes in a cluster so that the performance of the MAC protocol is not degraded.

The Hybrid Energy-Efficient Distributed clustering (HEED) [148] periodically selects cluster heads according to a hybrid of their residual energy and a secondary parameter, such as node proximity to its neighbors or node degree. HEED does not make any assumptions about the distribution or density of nodes, or node capabilities. A careful selection of the secondary clustering parameter can balance load among cluster heads and distribute the cluster heads well. The HEED protocol terminates in a constant number of iterations, independent of the network diameter, but it only generates 1-hop clusters, which may be suitable only for networks with a small number of nodes.

The Distributed Clustering Algorithm (DCA) uses weights associated with nodes to elect cluster heads [26]. These weights are generic and can be defined based on the application. It elects the node that has the highest weight among its 1-hop neighbors as the cluster head. The DCA algorithm is suitable for networks in which nodes are static or moving at a very low speed. The Distributed and Mobility-Adaptive Clustering Algorithm (DMAC) modifies the DCA algorithm to allow node mobility during or after the cluster set-up phase [25]. However, both algorithms generate 1-hop clusters, require synchronized clocks and have a complexity of $O(n)$. This makes them suitable only for networks with a small number of nodes.

SPAN [35] and GAF [139] are geographic-topology based protocols that utilize location information to eliminate unnecessary links. In GAF, geographic location is assumed to be available based on a positioning system such as GPS, while SPAN infers this information through broadcast messages and routing updates. However, the position of the nodes in practice is usually difficult or costly to know.

LEACH [60] is an energy-efficient clustering protocol in wireless sensor networks. It utilizes randomized rotation of cluster heads to evenly distribute the energy load among sensors in

the network and achieves dynamic clustering by choosing a cluster head that requires the minimum communication energy to each node within the cluster. The algorithm is run periodically, and the probability of becoming a cluster head for each period is chosen to ensure that every node becomes a cluster head at least once within $1/P$ rounds, where P is the desired percentage of cluster heads. This ensures that none of the sensors are overloaded because of the added responsibility of being a cluster head. However, all the nodes need to be time-synchronized first, which by itself is a challenging task. DRC does not need time-synchronize all the nodes. LEACH allows only 1-hop clusters to be formed, which might lead to a large number of clusters. Moreover, LEACH is designed for data-aggregation applications, where data collected from different sources are combined and aggregated through a network hierarchy and finally reach the processing sensor. In these kinds of applications, all nodes need to participate in the cluster formation process, whether there is an event or not.

CMLDA [40] is a data-collection algorithm that focuses on how to find an efficient manner in which the data should be collected from all the sensors and transmitted to the base station, such that the system lifetime is maximized. It uses clusters to maximize the system lifetime and demonstrates that the lifetime of a schedule given by the CMLDA heuristic is always within 10% of the optimal fractional solution. While CMLDA only considers system lifetime, DRC also considers energy usage.

Estrin et al. presented a new multi-level localized clustering method as part of the direct diffusion protocol in [45]. In their algorithm, they associate sensors at a particular transmission power level with a radius. The radius specifies the number of physical hops that a sensor's advertisements will travel. By increasing the hop number, the sensors can increase the transmission range and thus find a proper cluster head to form clusters. Their approach includes three key steps: *advertisement period*, *wait period*, and *promotion period*. All sensors start off at level 0, periodically sending out advertisements to sensors that can be reached within a certain number of hops. Sensors also start a *wait timer* afterward to make sure the advertisements

have been propagated. At the end of the wait period, the sensors then start a *promotion timer* if they do not have a parent. The promotion timer is set to be inversely proportional to the sensors remaining energy and the number of other sensors from whom level 0 advertisements were received. When the promotion timer expires, a sensor promotes itself to a higher level if no parent is found and starts sending periodic advertisements at that level of radius. However, this clustering protocol does not consider events occurring in wireless sensor networks and so is a proactive protocol. After forming the clusters, a node may have to communicate with a cluster head far away from phenomena, which is not energy efficient.

5.2 Detail Descriptions of DRC

DRC achieves three objectives: First, it is reactive. It saves energy by only waking up the nodes that detect the event, while other nodes can still stay in the sleep mode. Second, it uses power-control techniques to minimize the transmission power used for communication between two nodes. Third, it is a localized clustering protocol that via simple local node interactions achieves a desired global objective. The advantages of a localized protocol also lies in its good scalability and robustness, which are particularly important to wireless sensor networks [45].

The operation of DRC begins with a *post-deployment phase*, followed by a *cluster forming phase* which is when the clusters are constructed, then an *intra-cluster data processing phase*, and finally a *cluster head to processing center phase*. We assume there is no inter-cluster data transmission after clusters are formed.

A sensor node executing the DRC protocol must stay in one of the following states, which can be differentiated using the node-coloring approach [41]:

1. Gray: Unclustered node.
2. Black: Cluster head.
3. White: Participating nodes within a cluster.

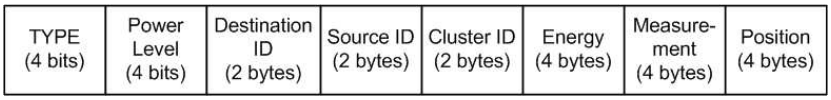


Figure 5.2: Message format.

DRC uses several messages to exchange information among nodes. Each message is 15 bytes long with 7 fields, as shown in Fig. 5.2.

- "TYPE" indicates the type of the message which can be REQUEST, REPLY, JOIN, JOIN-FORWARD, BEACON, CHANGE-PHASE and SLEEP. REQUEST is the message sent out by an unclustered node to form a cluster; REPLY is the message replying the REQUEST; JOIN is the message indicating the sending node wants to join an existing cluster; JOIN-FORWARD is used to forward a JOIN message to the cluster head; The CHANGE-PHASE message is to change the phase of DRC; and SLEEP is a message to put the receiving nodes into the *sleep* mode.
- "Power Level" is the transmission power the node currently uses. We assume by using the highest power level, every node can reach the processing center.
- "Destination ID" is the destination node identification and we use all 1's to indicate broadcasting.
- "Source ID" is the identification of the current node.
- "Cluster ID" is the identification of the cluster head the current node belongs to and we use all 0's to indicate that the current node is unclustered.
- "Energy" is the amount of remaining energy of the node.
- "Measurement" is the target signal measured by the current node at current moment.

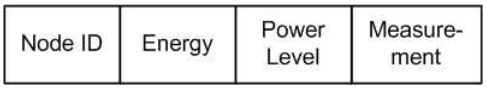


Figure 5.3: Neighbor table.

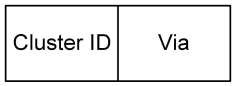


Figure 5.4: Routing table.

- "Position" is the sensor node's position.

Each node maintains two tables: a *neighbor table* and a *routing table*. The neighbor table, as shown in Fig. 5.3, stores the neighboring node information. The neighbor table is periodically updated after the node receives BEACON message, which contains the most up-to-date information of the neighbor node. The neighbor table will be used by the upper layer when the mobile agent needs to decide which node it should migrate to. The routing table, as in Fig. 5.4, is a table that stores the neighbors' node ID via which the node can send message to the cluster head.

For the cluster head, besides these two tables, DRC also has a *participation table* recording information of the participating nodes in the cluster and via which node the cluster head can send message to these nodes; see Fig. 5.5.

5.2.1 Post-Deployment Phase

we assume the node deployment is random. At this stage, all nodes turn off their radios and put the CPUs into the "sleep" mode, with only the sensor left functioning. All the nodes are assigned color Gray (unclustered).

Node ID	Via

Figure 5.5: Participation table.

5.2.2 Cluster Forming Phase

When an event occurs (e.g. a target moves inside the sensor field), the nodes within a certain range of that event will detect it and begin to collect data. In the meanwhile, they will wake up their radio and CPU. The initial transmission power is set to the lowest level of the m available power levels. These nodes will wait for T_x amount of time before they can send out a broadcast message to form clusters. The reason for this is to wait for an incoming message which would make the broadcast message unnecessary. According to [47], radio transmission takes the most power and the nodes with less energy should avoid excessive transmissions. During this period, the nodes can receive messages, such as REQUEST, REPLY, and JOIN, but cannot send a message. We model the T_x on node *x* using a zero mean Gaussian:

$$T_x = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{e_x^2}{2\sigma^2}} \tag{5.1}$$

where e_x is the available energy on node x and σ is the standard deviation, a parameter determines how fast T_x decreases when e_x increases. In other words, the waiting time T_x is related to the remaining energy of the current node in a Gaussian sense in that the nodes with more available energy will send out messages earlier than those with less energy and thus are more likely to become cluster heads. Using this relation can save the energy consumption of the nodes with less available energy and meet our requirement of prolonging the network lifetime in wireless sensor networks.

If a node does not receive any message before T_x expires, it will broadcast a REQUEST message, and start another timer T_{wait}. T_{wait} is the waiting time before retransmission and it is the same on all nodes. If no message is received before T_{wait} expires, it will increase its transmission power to the next available level and broadcast the REQUEST message again. The transmission range of that node will be increased accordingly when the transmission power increases. We assume a node can transmit with *m* discrete power levels, with P_m being the largest level. If the node reaches the P_m level but still cannot receive any message from other nodes, it will elect itself as the cluster head.

There may be a situation that both node A and B send out REQUEST messages simultaneously and both receive the REQUEST message sent from the other. The cluster head election procedure will ensure that both node A and B will choose the same cluster head and join the same cluster. The procedure proposed is described here: The node first compares its available energy with the "Energy" field in the received REQUEST message and chooses the bigger one as the cluster head. If they are the same, it then compares each others' "Information Gain" value and chooses the larger one as the cluster head.

Assume node A broadcasts a message to a group of nodes within its transmission range and also assume node B is one of them, then there are totally four different scenarios we need to consider:

- Scenario 1: Both A and B are unclustered and node A initializes the process by sending out the REQUEST message.
- Scenario 2: Node A is clustered but node B unclustered. Node A sends out REPLY or JOIN message and node B overhears the message.
- Scenario 3: Node A is unclustered and node B clustered. Node A initializes the process by sending out a REQUEST message.
- Scenario 4: Both node A and B are clustered.

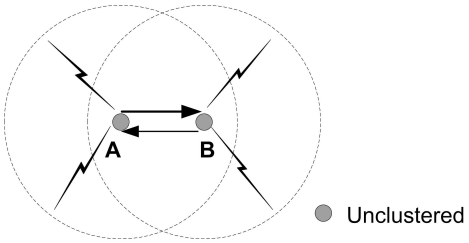


Figure 5.6: Scenario 1: Both nodes A and B are unclustered.

Scenario 1: Both nodes A and B are unclustered. (see Fig. 5.6). In this case, after an initial negotiation explained below, both nodes will assign themselves to the same cluster. The cluster head will be chosen as the node with the most remaining energy.

Node A first fills out the REQUEST message with the current transmission power level, node A's ID, current cluster ID as 0 (unclustered), remaining energy on board, and the information gain it measures. It then broadcasts this message.

Upon receiving this message, node B first compares its current transmission power with the power level recorded in the message. If its power level is less than that in the message, it will set its transmission power equal to the power level in the message in order to reach node A from node B. Here, we assume that the fall-off of transmission strength is uniform and symmetric. That is, if a sender uses a certain level of transmission power to reach a receiver, the receiver can also use the same transmission power to reach the sender. If the node's transmission power is higher than that in the message, it will not lower its transmission power accordingly since it still needs to reach its other neighbors. Node B then adds one entry in the neighbor table describing node A's information, such as the available energy and information gain. It will also add one entry in the routing table indicating via which node the message can reach the cluster head. If node B has a higher available energy, it will elect itself as the cluster head following the cluster head election procedure described above. It then changes its color to either Black or White accordingly. Based on the fact if node A is selected as cluster head or not, node B will

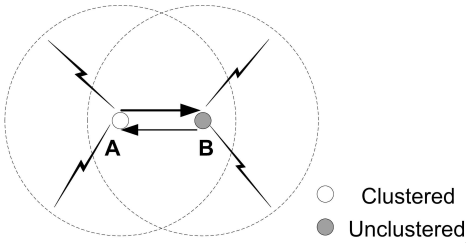


Figure 5.7: Scenario 2: Node A belongs to a cluster, node B is unclustered.

fill out a JOIN or REPLY message, indicating its own available energy, information gain, and the cluster ID as the cluster head ID, and broadcast the message to A. If node B is selected as the cluster head, it will add one entry of B's participation table.

Once node A gets the message from B, it can modify its neighbor table accordingly. If node B is chosen as the cluster head, node A will change its color to White and then broadcast a JOIN message. All the nodes that choose node A as the cluster head will change their membership to node B when receiving this message. Similarly, if node A is the cluster head, it will fill in the participation table and change its color to Black.

Scenario 2: If node A belongs to a cluster, and node B is unclustered, node B will join the cluster that node A belongs to, as Fig. 5.7 shows.

Node A uses the same procedure as described in Scenario 1 to broadcast the REPLY or JOIN message except that now the cluster head area in the message is the cluster head ID.

For node B, after the same procedure to modify the neighbor table and the routing table, it will broadcast a JOIN message.

When node A receives the JOIN message, it modifies it to a JOIN-FORWARD message and sends it to the node in its routing table that routes to the cluster head. It is a point-to-point transmission because node A need not broadcast the forwarding message.

Scenario 3: If node A is unclustered and node B belongs to a cluster, node A will join the cluster that node B belongs to, as Fig. 5.8 shows.

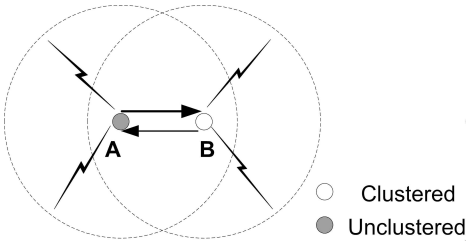


Figure 5.8: Scenario 3: Node A unclustered, node B belongs to a cluster.

The request sending procedure for node A is exactly the same.

For node B, it will send a reply message back to A indicating its cluster ID. If node B is not the cluster head, it also sends a JOIN-FORWARD message for node A by filling in the node ID area as node A's ID, and the destination node is the node in its routing table that goes back to the cluster head. It is also a point-to-point transmission.

Once node A receives the REPLY message from node B, it will modify the neighbor table and routing table accordingly.

Scenario 4: If node A and B belong to different clusters, node B will discard the message, as Fig. 5.9 shows.

Since there is no inter-cluster data transmission in DRC, the nodes belonging to different clusters will be "invisible" to each other. So node B will discard the message from node A, and thus is "invisible" to node A.

Once selected as a cluster head, it waits for a certain amount of time (several seconds) before sending all its members a CHANGE-PHASE message, indicating the end of the cluster-forming phase. The cluster head keeps a participating table and it will add an entry when it receives either a JOIN or JOIN-FORWARD message. After the cluster forming phase is ended, the cluster enters the intra-cluster data processing phase, during which all the cluster forming messages will be ignored.

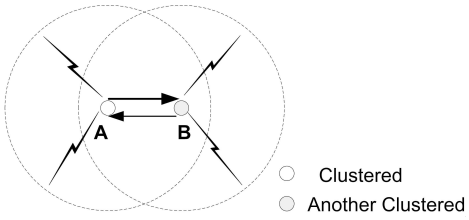


Figure 5.9: Scenario 4: Node A and B belong to different clusters.

5.2.3 Intra-Cluster Data Processing Phase

Once the nodes receive the CHANGE-PHASE message, they will enter the intra-cluster data processing phase. If the upper layer chooses the client/server computing paradigm, the nodes will transmit their data to the cluster head. We use a contention -ased MAC protocol to avoid the need for time synchronization. The cluster head will generate a partial result after receiving the data from the nodes. The nodes continue to send out their data until no events are detected, when they will go back to the sleep mode. If the upper layer chooses the mobile agent computing paradigm, a mobile agent will be sent out from the cluster head and migrates within the cluster. Once a sensor node receives the mobile agent, it will go back to the sleep mode. When the mobile agent has a satisfying result or has no sensor nodes left to migrate to, it will return to the cluster head. Once the cluster head generates the partial result, the intra-cluster data processing phase ends and the cluster-head-to-processing-center phase begins. In order to indicate the functionality of the sensor nodes and wireless links, and update the information on the neighbor node, each sensor node periodically sends out BEACON message. Once a sensor node receives a a BEACON message, it will update its neighbor table.

5.2.4 Cluster-Head-to-Processing Center Phase

In this phase, if the upper layer chooses the client/server computing paradigm, the cluster heads will increase their transmission powers to the highest level and send the partial results to the

processing center. We also use a contention based MAC protocol to avoid collisions. Otherwise, a mobile agent will be dispatched by the processing center to migrate among the cluster heads. The cluster heads then go back to the sleep mode; this concludes the procedure of DRC. Similarly, each cluster head periodically sends out a BEACON message to guide the migration of the mobile agent.

Fig. 5.10 shows the clusters after running DRC when the nodes detect an acoustic event (a target moving through the network with a speed of 20 m/s). The area of the wireless sensor network is 100 m $\times$ 100 m. The black circles in the figure represent the cluster heads which communicate with the processing center. The white circles represent member nodes in clusters. The gray circles represent sleeping nodes. The dashed line represents the track of the target. We can see that only those nodes that sense the event wake up and form the clusters. For comparison, Fig. 5.11 shows a proactive clustering of the same wireless sensor network with a transmission range of 15 m. There are much fewer connections than those in the predefined clustering network. Fig. 5.12 shows the network clustering when another acoustic event occurs with a different target movement. The clustering of the network adaptively changes according to the events.

5.3 Performance Evaluation

Although DRC possesses a couple of advantages, there are some potential problems with DRC as well. The DRC protocol causes overhead in forming the cluster. When the events are frequent, the overhead may become dominant. Therefore, it is necessary to evaluate the performance of DRC and compare it with other clustering protocols.

We have developed a simulator in JAVA to evaluate the performance of DRC. We also implement the LEACH protocol [60] and a predefined fixed clustering protocol to compare with DRC. DRC employs a power-control technique with 8 transmission power levels. The other two protocols only have one transmission power level with a transmission range equal to

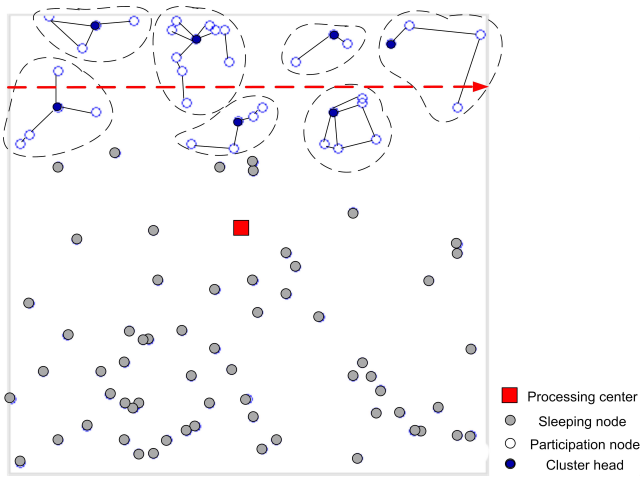


Figure 5.10: DRC clustering result after an event.

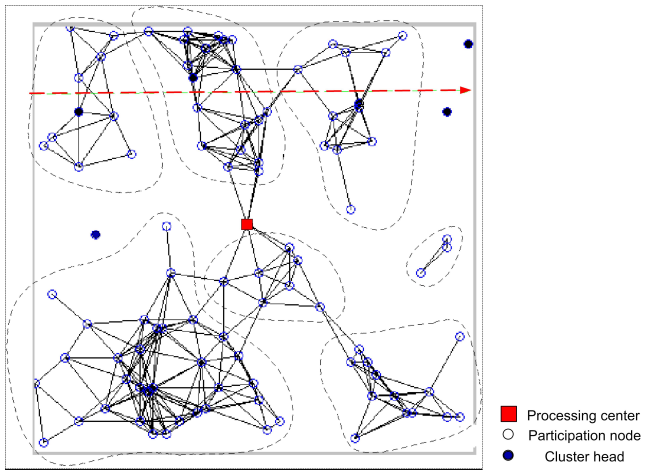


Figure 5.11: Predefined clusters.

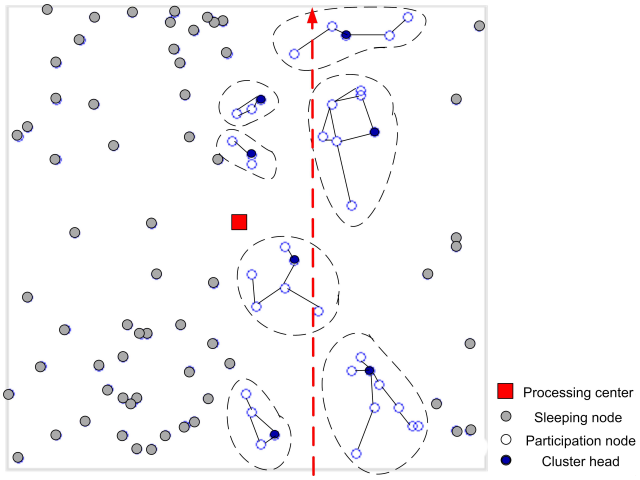


Figure 5.12: DRC result after another event.

Table 5.1: Power consumption in different states.

Mode	Power Consumption (mW)
Transmit	0.2355 - 52.981
Receive	10.50
Idle	10.36
Sleep	1.0

Table 5.2: Related parameter setup for the basic network.

No. of Nodes	Sensing range	Target speed	T_{wait}	No. of Events	Simulation time
100	20m	20m/s	0.1s	3	20s

75 m. The transmission power determines the transmission range by the following equation:

$$P_t = \alpha P_{r-thresh} d^2 \tag{5.2}$$

where $\alpha = \frac{4\pi^2}{G_t G_r \lambda^2}$, G_t, G_r are the antenna gain factors and are chosen to be 1, λ is the signal wavelength and is chosen to be 0.325 m, $P_{r-thresh}$ is the receiver threshold and is set to be 2.52×10^{-8} W. we set 8 transmission power levels: $P_1 = 0.2355$ mW, which determines a range of 5m, $P_2 = 2.119$ mW with a range of 15 m, etc. The maximum transmission power is $P_8 = 52.981$ mW with a range of 75 m. A node can be at transmit, receive, idle or sleep modes. The power consumption of each mode is listed in Table 5.1 according to [1].

We set up a basic network of 100 m $\times$ 100 m, where nodes can be *randomly* or *uniformly* deployed. The processing center is placed at the center of the field. The maximum distance between any nodes and the processing center is 70.7 m. Table 5.2 shows the parameters for the basic network. The number of events is the number of events occurred during the simulation time. The initial energy on all the nodes is 36 Joules, and the processing center does not have an energy limitation.

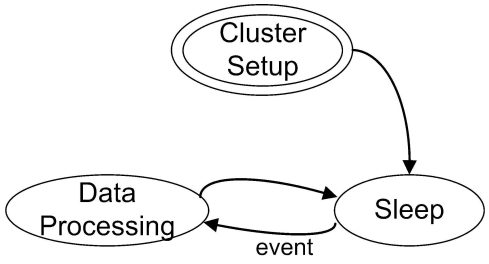


Figure 5.13: The operation of the fixed clustering protocol.

We use two metrics to compare the performance of different clustering protocols: the *energy consumption* and *network lifetime*. The *energy consumption* is defined as the total energy consumed by all the nodes in the wireless sensor network during the whole data processing procedure. The *network lifetime* is defined as the time from node deployment to the time when the first node is out of function because of energy depletion (assume all nodes start with the same fixed amount of energy).

For a fair comparison, we modified the LEACH protocol and added the reactive feature such that the nodes are only waken up by events. Furthermore, all protocols employ a 2-level client/server computing paradigm, where the nodes within a cluster will send the data they collect to the cluster head and then go to the sleep mode. After receiving the data, the cluster head will generate a partial result and then send it to the processing center. For the fixed clustering protocol, we predefine a four-cluster network with a node transmission range of 75 m. This protocol only needs to set up the clusters once at the beginning. After that, the nodes can either participate in the data processing when there is an event or stay in the sleep mode. Fig. 5.13 shows the timeline of its operation. The LEACH and DRC protocols will dynamically change their clusters. LEACH changes its clustering periodically, while DRC is driven by events. See Fig. 5.14 for their operations.

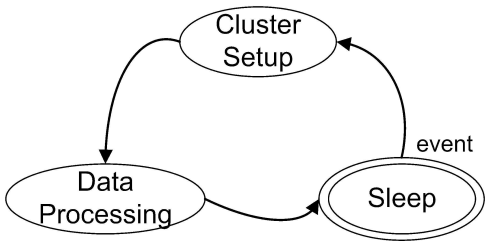
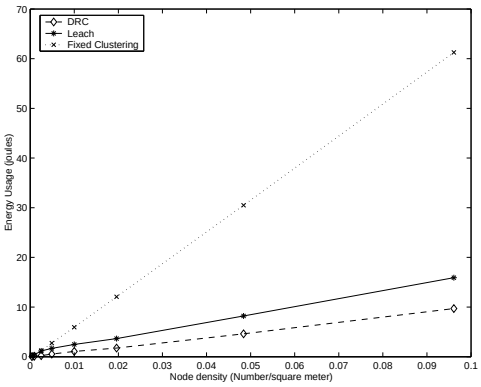


Figure 5.14: The operation of the LEACH and DRC protocols.

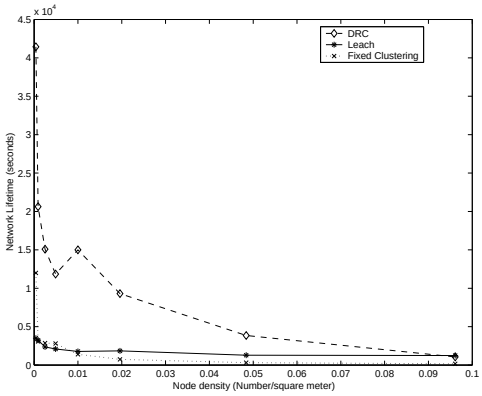
We designed four experiments to evaluate the effect of different parameters on the protocol performance. These parameters include the number of nodes, target speed, signal range (how far away a target signal can be captured by a sensor), and the number of events. We used two metrics: *energy consumption* and *network lifetime* for the evaluation. In each experiment, we changed one of the parameters of the basic network and kept all the others unchanged.

5.3.1 Effect of the Node Density

In this experiment, we uniformly deploy different numbers of nodes in a 100 m $\times$ 100 m sensor field. From Fig. 5.15(a) we can observe that the total energy consumption for all three protocols increases as the node density increases. However, the fixed clustering protocol grows much faster than the other two protocols. This is because the nodes within clusters have to communicate with the cluster head using the largest transmission power available as other nodes in the cluster may be in the sleep state and cannot act as a router for the wake-up nodes. This will consume more energy, especially at the time when the number of nodes is large and the nodes are densely deployed in an area. We can also infer from the graph that the DRC always performs better than LEACH, which shows the advantage of reactive clustering. In term of network lifetime shown in Fig. 5.15(b), DRC exhibits the longest lifetime as compared to LEACH and fixed clustering protocols.

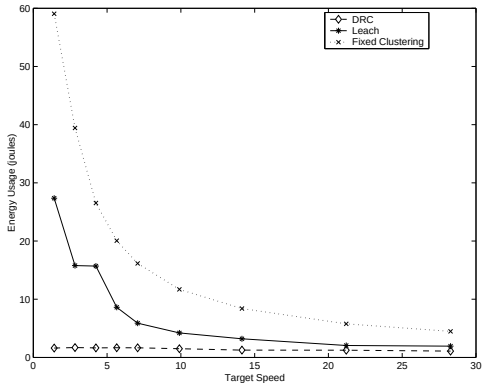


(a) Total energy consumption.

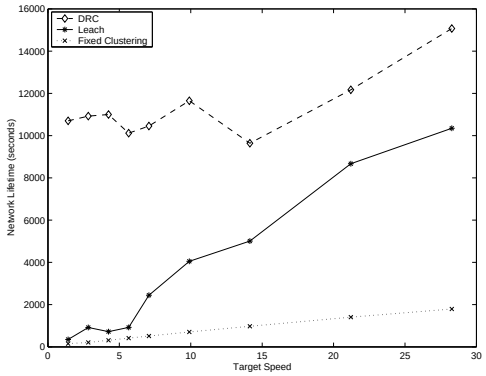


(b) Network lifetime.

Figure 5.15: The effect of node density.



(a) Total energy consumption.



(b) Network lifetime.

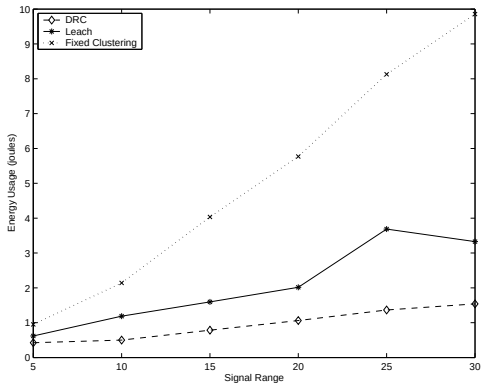
Figure 5.16: The effect of the target speed.

5.3.2 Effect of the Target Speed

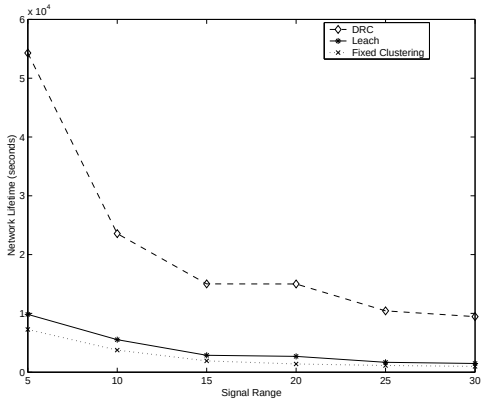
In this experiment, we fix the node number at 100 and observe the effect of target speed on the performance of different protocols. We expect a decrease of energy consumption as target speed increases because the time that the nodes stay within the signal range of the target will be shortened as the target speed increases. We can see from Fig. 5.16(a) that DRC always consumes less energy than the other two protocols. Fig. 5.16(b) shows the lifetime of the network. We observe, similarly, that the network lifetime increases as the target speed increases.

5.3.3 Effect of the Signal Range

In this experiment, we change the signal range of the target from 5 m to 30 m and keep all the other parameters in the basic network unchanged. We observe from Fig. 5.17(a) that the energy usage grows as the signal range increases. This is because more nodes will detect a target and thus more node wake-ups. However, no matter how large the signal range is, DRC always performs the best. For the same reason, we observe from Fig. 5.17(b) that the lifetime



(a) Total energy consumption.



(b) Network lifetime.

Figure 5.17: The effect of the signal range.

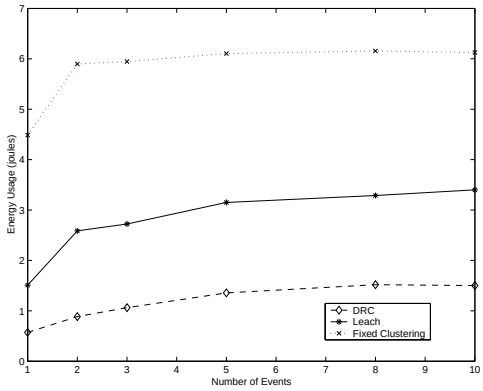
decreases steadily as the signal range increases, but DRC has a much longer lifetime than the other two.

5.3.4 Effect of the Number of Events

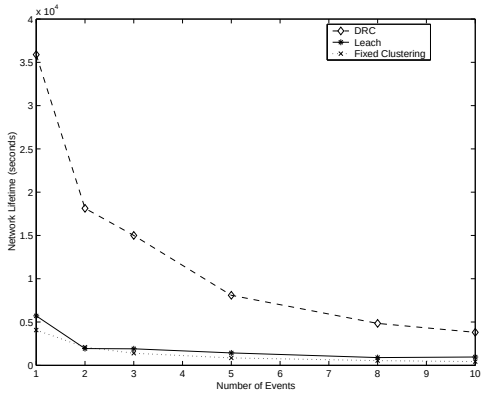
In this experiment, we observe the effect of the number of events by changing it from 1 to 10 and keeping all the other parameters in the basic network unchanged. We find from Fig. 5.18(a) that the energy usage grows as more events occur. This is because more events would cause more nodes to wake up and thus consume more energy. As to the lifetime of the network, it will decrease accordingly, as shown in Fig. 5.18(b).

5.4 Discussions

In order to combat the energy constraint problem of the sensor network, we propose a reactive clustering method called Decentralized Reactive Clustering (DRC), which has several desirable features: First, it is an event-driven protocol that forms clusters reactively. That is, only those



(a) Total energy consumption.



(b) Network lifetime.

Figure 5.18: The effect of the number of events.

nodes close to phenomena take part in the clustering procedure while other nodes can still be in the sleep mode. Thus the energy usage of the entire network can be reduced. Second, it employs power control techniques to lower the energy usage. Several experiments have been designed to compare the performance of DRC, LEACH, and the fixed clustering protocols in terms of the energy consumption and network lifetime. A simulator was developed to evaluate the performance and the simulation results show that DRC outperforms the other two protocols and can greatly increase the energy efficiency and lifetime of the wireless sensor networks.

Chapter 6

Conclusions and Future Work

Previous chapters presented the author's work on energy-efficient designs in wireless sensor networks. This chapter summaries the contributions so far and discusses the directions of future work.

6.1 Summary of Contributions

The original contributions of this work include modeling and performance evaluation of the client/server and the mobile agent paradigm; the use of cluster-based computing model to combine the advantages of both paradigms; the modeling and design of information driven mobile agent planning problem; and the development of a new decentralized reactive cluster protocol to support the cluster-based computing paradigm.

6.1.1 Modeling and Performance Evaluation of Distributed Computing Paradigms

In order to support the collaboration among sensor nodes, a proper computing paradigm has to be developed that meets the requirements of CSIP in wireless sensor networks, such as energy efficiency, scalability, scalability, etc. We first present simulation models for both the

client/server paradigm and the mobile agent paradigm. The *execution time*, *energy* and *energy*delay* are used as metrics to measure the performance. Several experiments have been designed to show the effect of different parameters on the performance of different paradigms. Experimental results show that the mobile agent paradigm performs much better when the number of nodes is large while the client/server paradigm is advantageous when the number of nodes is small. A mobile agent framework has been developed in Python and successfully demonstrated in SensIT field demos for target classification and tracking.

6.1.2 Cluster-based Computing Paradigm

Based on the performance evaluation results, we further propose a cluster-based hybrid computing paradigm to combine the advantages of these two paradigms. There are four schemes in this paradigm and simulation results show that there is always one scheme which performs better than either the client/server or the mobile agent paradigm. Thus, the cluster-based hybrid computing provides an energy-efficient and high-performance solution to CSIP.

6.1.3 Mobile Agent Planning Modeling and Algorithms Design

The mobile agent migration route has a significant impact on the overall performance of the sensor network. We refer to this problem as the *Mobile Agent Planning* (MAP) problem. The MAP-related research can be divided into two branches, the Static Mobile Agent Planning (SMAP) and the Dynamic Mobile Agent Planning (DMAP). We first model the SMAP problem and derive the optimal itinerary for SMAP. Then we model the DMAP as an optimization problem. Based on the modeling, we propose three information driven mobile agent planning algorithms, ISMAP, IDMAP, and P-IDMAP. The advantage of P-IDMAP is that the algorithm can predict the movement of target for the next period of time and guide the mobile agent migration to follow the movement of the target in order to maximize the information gain and in the meantime minimize the energy consumption to prolong the lifetime of the sensor network. Three metrics,

energy consumption, network lifetime, and the number of hops, are employed to evaluate the algorithms. Simulation results confirm the advantages of P-IDMAP algorithm for collaborative signal processing in wireless sensor networks.

6.1.4 Decentralized Reactive Clustering

Since the sensor platforms are usually densely deployed in an environment of interest, which induces a high level of redundancy, it is more efficient from an energy conservation point of view if only a subset of these sensors stay active at a certain time. This way, the sensor network can use its resources more effectively and last longer. Self-clustering is a good approach to achieve this, in which sensor platforms within a certain neighborhood form a cluster and a cluster head is automatically selected. Communications only occur between sensor platforms within the same cluster or between cluster heads. The underlying clustering protocol should also provide aids for CSIP using mobile agent computing paradigm and guide the migration of the mobile agent. Previous clustering protocols are proactive, which may not be suitable for the event-driven CSIP in sensor networks. We thus propose a decentralized reactive clustering (DRC) protocol where the clustering procedure is initiated only when events are detected. It uses power control technique to minimize energy usage in forming clusters. Simulation results show considerable improvements over LEACH in energy conservation and network lifetime using DRC.

6.2 Directions for Future Work

The ideas and contributions in this dissertation have a wide application to collaborative signal and information processing in wireless sensor networks. Several interesting future research directions are suggested here.

6.2.1 Cross-layer Optimization for CSIP in Wireless Sensor Networks

Since all layers of the protocol stack contribute to the energy consumption and delay for the CSIP applications, an efficient CSIP system requires a joint design across all these layers as well as the underlying hardware where the energy is actually expended [55]. Cross-layer design is particularly important for wireless sensor networks, since the state of the physical medium can significantly vary over time. Layers can exchange information to make more optimal usage of the network. It is beneficial and feasible to further reduce the energy consumption of the mobile agent migration utilizing the cross-layer protocol design and optimization.

6.2.2 Multi-Agent System for CSIP

In our mobile agent planning modeling and algorithm design, we only consider the case of single mobile agent migration. It is a much complex problem for multi-agent migration. In multi-agent systems, the mobile agent has the ability to be social and to interact with other mobile agents. In order for agents to interact, they must possess the ability to communicate with other agents via some common *agent communication language* (ACL). For the CSIP applications, mobile agents need to cooperate with each other from time to time to exchange information, guide migration itinerary, and pursue the common goal.

6.2.3 Strong Migration of Mobile Agent Framework

We have implemented a mobile agent hierarchy in Python and C++. During the mobile agent migration process, only the executable code and member are migrated, and therefore only forms a weak migration. It is desirable to have a strong migration mechanism, in which not only code and the data member, but the execution environment need to be migrated as well. In order to achieve strong migration, we need to design a method to migrate current stack and program counter either at the source code level or at the byte code and interpreter level.

6.2.4 Distributed Data Mining in Wireless Sensor Networks Using Mobile Agents

One challenge of wireless sensor networks is it huge amount of data need to be processed. These data are usually distributed in a diverse and geographically dispersed environment. It is essential to find useful information and discover knowledge from these data. Data mining is a technology that deals with the discovery of hidden knowledge, unexpected patterns and new rules from large data set. However, current data mining systems are mainly centralized, not suitable for the distributed environment, such as the sensor network. It is desirable have a mobile-agent framework to facilitate distributed data mining, where mobile agents are dispatched from node to node to perform distributed data mining algorithms. The advantages of the mobile-agent based data mining system over the client/server-based data mining system will be studied in terms of the energy consumption, network lifetime, and scalability and reliability.

6.3 Publication History

This dissertation appears in part in the following academic journals and conferences.

Book Chapters

- H. Qi, **Y. Xu**, P. T. Kuruganti (2004). "Chapter 41: The Mobile Agent Framework for Collaborative Processing in Sensor Networks", *Distributed Sensor Networks*, Editor: R. Brooks, S. S. Iyengar, pages 783-800, CRC Press.

Refereed Journals

- **Y. Xu**, H. Qi (2004). "Distributed Computing Paradigms for Collaborative Signal and Information Processing in Sensor Networks", *International Journal of Parallel and Distributed Computing*, 64(8): 945-959, August.

- H. Qi, **Y. Xu**, and X. Wang (2003). "Mobile-agent-based collaborative signal and information processing in sensor networks", *Proceedings of IEEE, Special Issue on Sensor Networks and Applications*, Volume: 91, Issue: 8, Pages: 1172 - 1183, August.

- H. Qi, P. T. Kuruganti, **Y. Xu** (2002). "Collaborative signal and information processing hierarchy in distributed sensor networks", *Sensors Journal*, 2(7): 270-285, July.

Conferences

- **Y. Xu**, H. Qi (2004). "Decentralized Reactive Clustering for Collaborative Processing in Sensor Networks", *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 54-61, Newport Beach, USA, July.

- **Y. Xu**, H. Qi, P. T. Kuruganti (2003). "Computing Paradigms for Collaborative Processing in Sensor Networks" *IEEE 2003 Global Communications Conference (GLOBECOM)* , Volume: 6, Pages: 3531-3535, San Francisco, USA, December.

- **Y. Xu**, H. Qi (2002). "Performance evaluation of distributed computing paradigms in mobile ad hoc sensor networks", *International Conference on Parallel and Distributed Systems (ICPADS)*, pages 451-456, Taiwan, December.

Bibliography

Bibliography

[1] ASH transceiver's designers guide. RF Monolithics, Inc. http://www.rfm.com.

[2] AVR 8-bit RISC CPU. Atmel Corporation. http://www.atmel.com/atmel/products/prod23.htm.

[3] http://www.controlled.com/pc104faq/pc104.

[4] http://www.isi.edu/scadds/pc104testbed.

[5] IVY - A sensor network infrastructure for the college of engineering. http://www-bsac.eecs.berkeley.edu/projects/ivy/.

[6] μ-adaptive multi-domain power aware sensors. http://www-mtl.mit.edu/research/icsystems/uamps/research/overview.shtml.

[7] ObjectSpace Voyager. Recursion Software, Inc. http://www.objectspace.com/products/prodVoyager.asp.

[8] Roc analysis: Web-based calculator for roc curves. http://www.rad.jhmi.edu/jeng/javarad/roc/main.html.

[9] Sensor information technology. Defense Advanced Research Projects Agency. http://www.darpa.mil/ito/research/sensit.

[10] SH-4 32-bit RISC CPU core family. SuperH, Inc. http://www.superh.com/products/sh4.htm.

[11] The Spec node. University of California, Berkeley. http://www.jlhlabs.com/jhill_cs/spec/.

[12] Talk II - SINCGARS multiservice communications procedures for the single-channel ground and airborne radio system. http://www.fas.org/man/dod-101/sys/land/docs/sincgars.pdf.

[13] TinyOs. University of California, Berkeley. http://webs.cs.berkeley.edu/tos/.

[14] WINSNG 2.0 user manual and API specification. Sensoria Corporation. http://www.sensoria.com/.

[15] FIPA 1997 version 2.0 specifications. Available at http://www.fipa.org/spec/fipa97.html, 1997.

[16] J. Agre and L. Clare. An integrated architecture for cooperative sensing networks. *IEEE Computer Magazine*, pages 106–108, May 2000.

[17] J. R. Agrea, L. P. Clarea, G. J. Pottieb, and N. P. Romanova. Development platform for self-organizing wireless sensor networks. volume 3713, pages 257–268, 1999.

[18] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, August 2002.

[19] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38:393–422, 2002.

[20] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM*, January 2000.

[21] H. Attias. Inferring parameters and structure of latent variable models by variational Bayes. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 21–30, 1999.

[22] J. Baek, G. Kim, and H. Yeom. Timed mobile agent planning for distributed information retrieval. In *Int'l Conference on Autonomous Agent*, May 2001.

[23] J. Baek, J. H. Yeo, and H. Y. Yeom. Agent chaining: An approach to dynamic mobile agent planning. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, 2002.

[24] D. J. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Transactions on Communications*, 29(11):1694–1701, November 1981.

[25] S. Basagni. Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks. In *Proceedings of Vehicular Technology Conference,*, volume 2, pages 889–893, 1999.

[26] S. Basagni. Distributed clustering for ad hoc networks. In *Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks*, pages 310–315, June 1999.

[27] P. Bauer, M. Sichitiu, R. istepanian, and K.Premaratne. The mobile patient: Wireless distributed sensor networks for patient monitoring and care. In *Proceedings Of the First Annual IEEE Conference on Information Technology Applications in Biomedicine*, pages 17–21, 2000.

[28] J. Baumann, F. Hohl, K. Rothermel, and M. Straber. Mole - concepts of a mobile agent system. *World Wide Web*, 1(3):123–137, 1998.

[29] L. Benini and G. DeMicheli. *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Academic Publishers, 1997.

[30] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan amd Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[31] R. R. Brooks, C. Griffin, and D. S. Friedlander. Self-organized distributed sensor network entity tracking. *The International Journal of High Performance Computing Applications*, 16(3):207–219, Fall 2002.

[32] D. Cerpa and D. Estrin. ASCENT:adaptive self-configuring sensor networks topologies. In *INFOCOM*, 2002.

[33] A. P. Chandrakasan and R. W. Brodersen. *Low Power CMOS Digital Design*. Kluwer Academic Publishers, 1996.

[34] M. Chatterjee, S. K. Das, and D. Turgut. WCA: A weighted clustering algorithm for mobile ad hoc networks. *Journal of Cluster Computing, Special issue on Mobile Ad hoc Networking*, (3):193–204, 2002.

[35] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks Journal*, September 2002.

[36] Y. P. Chen and A. L. Liestman. A zonal algorithm for clustering ad hoc networks. *International Journal of Foundations of Computer Science*, 2003.

[37] C. F. Chiasserini, I. Chlamtac, P. Monti, and A. Nucci. Energy efficient design of wireless ad hoc networks. In *Proceedings of European Wireless*, February 2002.

[38] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. Technical report, Xerox Palo Alto Research Center, 2001.

[39] A. D. Costa and A. M. Sayeed. Collaborative signal processing for distributed classification in sensor networks. In *The 2nd International Workshop on Information Processing in Sensor Networks*, Palo Alto, CA, April 2003.

[40] K. Dasgupta, K. Kalpakis, and P. Namjoshi. An efficient clustering–based heuristic for data gathering and aggregation in sensor networks. In *proceedings of IEEE Wireless Communications and Networking Conference*, 2003.

[41] B. Deb, S. Bhatnagar, and B. Nath. A topology discovery algorithm for sensor networks with applications to network management. Technical Report DCS-TR-441, Rutgers University, May 2001.

[42] L. Doherty, B. A. Warneke, B. Boser, and K. S. J. Pister. Energy and performance considerations for smart dust. *International Journal of Parallel and Distributed Sensor Networks*, 4(3):121–133, Dec. 2001.

[43] J. Elson and D. Estrin. *Wireless Sensor Networks*, chapter Sensor Networks: A Bridge to the Physical World, pages 1–20. Kluwer Academic Publishers, 2004.

[44] D. Estrin, R. Govindan, and J. Heidemann. Embedding the internet. *ACM Communication*, 43:38–41, May 2000.

[45] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *MobiCom*, 1999.

[46] D. Estrin, A. Sayeed, and Srivastava. Tutorial on wireless sensor networks. In *Mobicom*, 2002.

[47] D. Estrin, A. Sayeed, and M. Srivastava. Mobicom 2002 tutorial: Wireless sensor networks. Mobicom 2002, 2002.

[48] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories, March 2004.

[49] T. Finin and R. Fritzson. KQML: A language and protocol for knowledge and information exchange. In *Proceedings of 19th International Distributed Artificial Intelligence Workshop*, pages 127–136, Seattle, USA, 1994.

[50] J. Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *Symposium on Operating Systems Principles*, pages 48–63, 1999.

[51] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding code mobility. *IEEE Trans. on Software Engineering*, 24(5):342–361, 1998.

[52] M. Gendreau, G. Laporte, and J. Y. Potvin. *Local Search in Combinatorial Optimizations*, chapter Vehicle routing: modern heuristics, pages 311–336. Wiley, 1997.

[53] C. Ghezzi and G. Vigna. Mobile code paradigms and technologies: a case study. In *Proceedings of the First International Workshop on Mobile Agents*, pages 39–49, Germany, April 1997.

[54] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh. Optimal energy aware clustering in sensor networks. *Sensors*, (2):258–269, 2002.

[55] A. J. Goldsmith and S. B. Wicker. Design challenges for energy-constrained ad hoc wireless networks. *IEEE Wireless Communications Magazine*, pages 8–27, Aug. 2002.

[56] R. S. Gray. Agent tcl: A transportable agent system. In *Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95)*, December 1995.

[57] L. J. Guibas. Sensing, tracking, and reasoning with relations. *IEEE Signal Processing Magazine*, pages 73–85, March 2002.

[58] Gregory Gutin and A. P. Punnen, editors. *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.

[59] C. G. Harrison, D. M. Chess, and A. Kershenbaum. Mobile agents: are they a good idea? Technical report, IBM Thomas J.Watson Research Center, March 1995.

[60] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *Proc. Hawaiian Int'l Conf. on Systems Science*, 2000.

[61] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, October 2002.

[62] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Mobicom*, 1999.

[63] A. Helal, B. Haskell, J. L. Carter, R. Brice, D. Woelk, and M. Rusinkiewicz. *Any Time, Anywhere Computing*. Kluwer Academic Publishers, 1999.

[64] B. E. Helvik and O. Wittner. Using the cross-entropy method to guide/govern mobile agent's path finding in networks. In *Proceedings of 3rd International Workshop on Mobile Agents for Telecommunication Applications*, pages 255–268, August 2001.

[65] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104, 2000.

[66] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[67] IEEE. Wireless medium access control (MAC) and physical layer (PHY) specification for low rate wireless personal area networks (LR-WPANS). IEEE 802.15.4-2003, 2003.

[68] T. Illmann, F. Kargl, M. Weber, and T. Kruger. Migration in java: Problems, classification and solutions. In *Proceedings of the MAMA'00*, Wollogong, Australia, December 2000.

[69] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, Boston, Massachusetts, August 2000.

[70] C. Jaikaeo, C. Srisathapornphat, and C. Shen. Diagnosis of sensor networks. In *IEEE International Conference on Communications*, Helsinki, Finland, June 2001.

[71] N. R. Jennings and M. R. Wooldridge. *Agent Technology Foundations, Applications and Markets*. Springer-Verlag, 1998.

[72] D. Johansen, R. V. Renessea, and F. B. Schneider. Operating system support for mobile agents. In *Proceedings of the Fifth Workshop Hot Topics in Operating Systems (HotOS)*, pages 42–45, Washington, USA, 1995.

[73] D. B. Johnson and D. A. Maltz. *Mobile Computing*. Kluwer Academic Publishers, 1996.

[74] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for smart dust. In *ACM Mobicom*, pages 271–278, Seattle, Washington, August 1999.

[75] S. Kumar, D. Shepherd, and F. Zhao. Collaborative signal and information processing in micro-sensor networks. *IEEE Signal Processing Magazine*, 19(2):13–14, March 2002.

[76] P. T. Kuruganti. Development of mobile agent framework in wireless sensor networks for multi-sensor collaborative processing. Master's thesis, University of Tennessee, 2003.

[77] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1998.

[78] S. Lee, A. Ermedahl, S. L. Min, and N. Chang. Statistical derivation of an accurate energy consumption model for embedded processors. Submitted to special issue on Power-Aware Embedded Computing: The Role of Static, Dynamic, and Adaptive Techniques.

[79] D. Li, K. D. Wond, Y. H. Hu, and A. M. Sayeed. Detection, classification, and tracking of targets. *IEEE Signal Processing Magazine*, 19(2):17–29, March 2002.

[80] C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *Journal on Selected Areas in Communication*, 15:1265–1275, September 1997.

[81] S. Lindsey and C. Raghavendra. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9), Sep. 2002.

[82] J. Liu, J. Reich, and F. Zhao. Collaborative in-network processing for target tracking. *Journal on Applied Signal Processing*, pages 378–391, March 2003.

[83] J. Liu, F. Zhao, and D. Petrovic. Information-directed routing in ad hoc sensor networks. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 88–97, 2003.

[84] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He. RAP: A real-time communication architecture for large-scale wireless sensor networks. In *Real-Time Technology and Applications Symposium*, September 2002.

[85] S. R. Madden, M. J. Franklin, J. M. Hellerstain, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *USENIX 5th Symp. Operating Syst. Design Implementation*, Boston, December 2002.

[86] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor network for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[87] D. Milojicic. Trend wars - mobile agent applications. *IEEE Concurrency*, 7(3):80–90, July 1999.

[88] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. MASIF: The OMG mobile agent system interoperability facility. In *In Proceedings of Mobile Agents*, Sept. 1998.

[89] R. Min, T. Furrer, and A. Chandrakasan. Dynamic voltage scaling techniques for distributed microsensor networks. In *Proceedings of the IEEE Computer Society Annual Workshop on VLSI*, page 43, 2000.

[90] K. Moizumi and G. Cybenko. The traveling agent problem. Technical report, Dartmouth College, Feb. 1998.

[91] K. Moizumi and G. Cybenko. The traveling agent problem. *Mathematics of Control, Signals and Systems*, January 1998.

[92] D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AoA. In *IEEE INFOCOM*, pages 1734–1743, San Francisco, 2003.

[93] T. A. Pering, T. D. Burd, and R. W. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *Proc. ISLPED*, pages 76–81, 1998.

[94] G. P. Picco. Mobile agents: an introduction. *Microprocessors and Microsystems*, 25:65–74, 2001.

[95] J. Polastre. Sensor network media access design. CS294-1 Fall 2003 Project Report, 2003.

[96] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):551–558, May 2000.

[97] J. G. Proakis and E. M. Sozer. Shallow water acoustic networks. *IEEE Communications Magazine*, November 2001.

[98] H. Qi, S. S. Iyengar, and K. Chakrabarty. Multi-resolution data integration using mobile agents in distributed sensor networks. *IEEE Trans. on Syst., Man, and Cybern. Part C: Applications and Reviews*, 31(3):383–391, Aug. 2001.

[99] H. Qi, X. Wang, S. S. Iyengar, and K. Chakrabarty. Multisensor data fusion in distributed sensor networks using mobile agents. In *Information Fusion*, August 2001.

[100] H. Qi, X. Wang, S. S. Iyengar, and K. Chakrabarty. High performance sensor integration in distributed sensor networks using mobile agents. *International Journal of High Performance Computing Applications*, 16(3):325–335, 2002.

[101] H. Qi, Y. Xu, and X. Wang. Mobile-agent-based collaborative signal and information processing in sensor networks. *Proceedings of the IEEE, Special Issue on Sensor Networks and Applications*, 91(8):1172–1183, Aug. 2003.

[102] J. M. Rabaey, M. J. Ammer, J. L. Silva, and S. Roundy. PicoRadio supports ad hoc ultra-low power wireless networking. *IEEE Computer Magazine*, 33:42–48, July 2000.

[103] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.

[104] V. Raghunathan, P. Spanos, and M. Srivastava. Adaptive power-fidelity in energy aware wireless embedded systems. In *IEEE real time systems symposium*, 2001.

[105] T. S. Rappaport. *Wireless Communications: Principles and Practice*. IEEE Press, 1996.

[106] S. Richardson and P. J. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society*, 59(4):731–758, 1997.

[107] S. Roberts and R. Everson, editors. *Independent Component Analysis: Principles and Practice*. Cambridge University Press, 2001.

[108] S. J. Roberts. Independent component analysis: source assessment and separation. *IEEE Proceedings on Vision, Image, and Signal Processing*, 145(3):149–154, 1998.

[109] A. Rosenstein, J. Li, and S. Y. Tong. MASH: The multicasting archie server hierarchy. In *ACM Computer Communication Review*, 1997.

[110] K. N. Ross, R. D. Chaney, G. V. Cybenko, D. J. Burroughs, and A. S. Willsky. Mobile agents in adaptive hierarchical bayesian networks for global awareness. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 2207–2212, 1998.

[111] J. Russell and M. Jacome. Software power estimation and optimization for high performance, 32-bit embedded processors. In *Proceedings of ICCD'98*, Oct. 1998.

[112] A. Savvides, S. Park, and M. B. Srivastava. On modeling networks of wireless micro sensors. In *SIGMETRICS*, June 2001.

[113] A. Savvides, M. Srivastava, L. Girod, and D. Estrin. *Wireless Sensor Networks*, chapter Localization in Sensor Networks, pages 327–349. Kluwer Academic Publishers, 2004.

[114] C. Schurgers, V. Tsiatsis, and M. Srivastava. STEM: Topology management for energy-efficient sensor networks. In *Proceedings of the IEEE Aerospace Conference*, 2002.

[115] J. R. Searle. *Speed Acts*. Cambridge University Press, Cambridge, UK, 1969.

[116] Sensoria Corporation. *Sensoria's Sensor Gateway Brochure*.

[117] C. Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor information networking architecture and applications. *IEEE Personal Communications*, pages 166–179, August 2001.

[118] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design fro energy-efficient wireless sensor networks. In *Seventh Annual ACM SIGMOBILE Conference on Mobile Computing and Networking*, July 2001.

[119] S. Singh and C. S. Raghavendra. PAMAS - power aware multi-access protocol with signaling for ad hoc networks. *ACM Computer Communication Review*, July 1998.

[120] A. Sinha and A.P. Chandrakasan. Jouletrack: A web based tool for software energy profiling. In *Proc. Design Automation Conf.*, pages 220–225, 2001.

[121] A. Sinha, A. Wang, and A. P. Chandrakasan. Algorithmic transforms for efficient energy scalable computation. In *Proc. ISLPED*, 2000.

[122] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE International Conference on Communications*, Helsinki, Finland, June 2001.

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

[124] M. Srivastava, R. Muntz, and M. Potkonjak. Smart kindergarten: Sensor-based wireless networks for smart developmental problem-solving environments. In *Proceedings of the ACM SIGMOBILE 7th Annual International Conference on Mobile Computing and Networking*, July 2001.

[125] F. Stann and J. Heidemann. RMST: Reliable data transport in sensor networks. In *Proceedings of the First International Workshop on Sensor Net Protocols and Applications*, Anchorage, Alaska, USA, April 2003. IEEE.

[126] N. P. Sudmann. TACOMA - fundamental abstractions supporting agent computing in a distributed environment. Technical report, Department of Computer Science, University of Tromso, Norway, November 1996.

[127] T. Sundsted. An introduction to agents. *Java World*, June 1998.

[128] C. Tang, C. S. Raghavendra, and V. Prasanna. Energy efficient adaptation of multicast protocols in power controlled wireless ad hoc networks. In *2002 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '02)*, May 2002.

[129] V. Tiwari, S. Malik, A. Wolfe, and M.T.C. Lee. Instruction level power analysis and optimization of software. *J. VLSI Signal Processing*, 13(2):1–18, 1996.

[130] G. Pottie V. Ailawadhi. An energy efficient mac protocol for Hanets (short paper).

[131] A. Wang, S-H. Cho, C. G. Sodini, and A. P. Chandrakasan. Energy-efficient modulation and MAC for asymmetric microsensor systems. In *Proc. ISLPED*, 2001.

[132] H. Wang, K. Yao, G. Pottie, and D. Estrin. Entropy-based sensor selection heuristic for target localization. In *IPSN'04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 36–45. ACM Press, 2004.

[133] X. Wang. *High accuracy distributed target detection and classification in sensor networks based on mobile agent framework*. Phd dissertation, University of Tennessee, December 2004.

[134] B. Warneke, B. Liebowitz, and K. S. J. Pister. Smart dust: Communicating with a cubic-millimeter computer. *IEEE Computer Magazine*, pages 2–9, January 2001.

[135] M. Weiser. The computer for the 21st century. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.

[136] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet. Concordia: An infrastructure for collaborating mobile agents. Technical report, Mitsubishi Electric ITA, Horizon Systems Laboratory, 1998.

[137] J. S. Wong and A. R. Mikler. Intelligent mobile agents in large distributed autonomous cooperative systems. *Journal of Systems and Software*, 47(2):75–87, 1999.

[138] Q. Wu, N. Rao, J. Barhen, S. S. Iyengar, V. K. Vaishnavi, H. Qi, and K. Charkrabarty. On computing mobile agent routes for data fusion in distributed sensor networks. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):740–753, June 2004.

[139] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the seventh annual international conference on Mobile computing and networking*, pages 70–84. ACM Press, 2001.

[140] Y. Xu and H. Qi. Performance evaluation of distributed computing paradigms in mobile ad hoc sensor networks. In *International Conference on Parallel and Distributed Systems (ICPADS)*, pages 451–456, December 2002.

[141] Y. Xu and H. Qi. Decentralized reactive clustering for collaborative processing in sensor networks. *International Conference on Parallel and Distributed Systems(ICPADS)*, pages 54–61, July 2004.

[142] Y. Xu and H. Qi. Distributed computing paradigms for collaborative signal and information processing in sensor networks. *International Journal of Parallel and Distributed Computing*, 64(8):945–959, August 2004.

[143] B. Yang, D. Y. Liu, K. Yang, and S. S. Wang. Strategically migrating agents in itinerary graph. In *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, November 2003.

[144] K. Yao, R. Hudson, C. Reed, D. Chen, and F. Lorenzelli. Blind beamforming on a randomly distributed sensor array system. In *Proceedings of IEEE JSAC*, volume 18, October 1998.

[145] F. Ye, A. Chen, S. Liu, and L. Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *Proceedings of Tenth International Conference on Computer Communcations and Networks*, pages 304–309, 2001.

[146] W. Ye, J. Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, June 2002.

[147] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad hoc routing for wireless networks. Technical report, University of Illinois at Urbana-Champaign, 2001.

[148] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *Proceedings of IEEE INFOCOM*, March 2004.

[149] M. Youssef, M. Younis, and K. Arisha. A constrained shortest-path energy-aware routing algorithm for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, March 2002.

[150] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *Proc. of the 12th workshop on Parallel and Distributed Simulations (PADS '98)*, pages 154–161, May 1998.

[151] X. Zeng, Y. Sun, and R. Shi. The code migration of mobile agents system. In *Technology of Object-Oriented Languages and Systems*, page 181, Beijing, China, September 1998.

[152] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.

[153] Y. Zou and K. Chakrabarty. Energy-aware target localization in wireless sensor networks. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications*, 2003.

Vita

Yingyue Xu was born in Shenyang, P. R. China. He received his BS and MS degrees in Electrical Engineering from Tianjin University, China in 1999 and 2001 respectively. In 2001, Yingyue enrolled into the doctoral program at the University of Tennessee in Electrical and Computer Engineering. At the same time, he joined the Advanced Imaging and Collaborative Information Processing group as a graduate research assistant where he completed his Doctor of Philosophy degree in 2005. He had been working on a 4-year project MU-FASHION (**Mu**lti-Resolution Data **F**usion using **A**gent-Bearing **S**ensors in **Hi**erarchically **O**rganized Distributed Sensor **N**etworks) sponsored by Defense Advanced Research Projects Agency, through SensIT (Sensor Information Technology) Program. Besides that, he had also worked on various projects related to ad hoc and sensor networks, such as vision-based mobile sensor platform design, energy-efficient underwater sensor network (USN), etc. He also worked as an intern to implement IEEE 802.15.4 MAC and Physical layer protocol. His major research areas are wireless sensor networks, distributed signal/image processing, and mobile agent computing paradigm in sensor networks.