A Comparison Of Agent Paradigms For Resource Management In Distributed Sensor Networks

Anish Anthony
University of Alabama at Birmingham

A COMPARISON OF AGENT PARADIGMS FOR RESOURCE MANAGEMENT IN DISTRIBUTED SENSOR NETWORKS

by

ANISH ANTHONY

A DISSERTATION

Submitted to the graduate faculty of The University of Alabama at Birmingham, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

BIRMINGHAM, ALABAMA

2007

A COMPARISON OF AGENT PARADIGMS FOR RESOURCE MANAGEMENT IN DISTRIBUTED SENSOR NETWORKS

ANISH ANTHONY

COMPUTER ENGINEERING

ABSTRACT

Management of power and other resources that effect field life is an important consideration in sensor networks. Sensing, data fusion, target tracking, and network resources are some of the factors that have to be effectively managed in order to improve the field life of an underwater distributed sensor network application. A hierarchical sensor network is considered in which the sensors report the range and bearing of the target to the cluster nodes. The cluster nodes then fuse data from the sensor nodes to generate a local estimate of the target position. The master node fuses data from the different cluster nodes to generate a global estimate of the target position. Intelligent agents present on the cluster and master nodes help to manage network resources. This dissertation presents and compares alternative agent paradigms applicable for resource management in the sensor network. The comparisons provide an increased understanding of how the performance of the sensor network changes with the number, functionality, and presence of agents at different levels in the network hierarchy.

A modular simulation framework based on object-oriented design was used in performance evaluation and to generate data for a detailed analysis of component interactions. The number of computations, number of communications, tracking performance, and intelligence quotient for different agent paradigms were compared.

The simulation framework developed in this dissertation demonstrated the scalability and flexibility of an agent-based system where task-specific agents can be added or removed to easily simulate different scenarios. An agent-based scenario in which multiple master nodes were present in the field and another scenario in which master and cluster agents were stacked on the same physical hardware achieved high field-life performance with acceptable tracking errors. Simulation results verify the effectiveness of using agents for resource management in sensor networks. The results comparing the different agent paradigms presented in this dissertation will assist designers of agent-based systems in using agents to their best advantage in scenarios similar to those explored here.

DEDICATION

This thesis is dedicated to my parents for their love, patience, and sacrifice.

ACKNOWLEDGEMENTS

Give thanks to the LORD, who is good, whose love endures forever. (Psalm 118:1)

This research would not have been possible without the help of a great number of people, to all of whom I am very grateful.

I would like to thank my advisor, Dr. Thomas C. Jannett, for his support, advice, and guidance throughout the research and writing phases of this dissertation and for funding me for all these years.

I would like to thank my Ph.D. committee members, Dr. B. Earl Wells, Dr. Dale Callahan, Dr. Gary Grimes, and Dr. Roy Koomullil, for their guidance and support.

I would like to thank Sandra Muhammad, Debbie Harris, and Maria Whitmire, the departmental staff, for all the help they have provided me during these years.

I would like to thank Dalton Nelson and Jon Marstrander, for their motivation and support, especially during the final stages.

I would like to thank my colleague, Parag Joshi, for his help, advice, and encouragement during these years. I would also like to thank Shripad Chandrachood and Ramesh Praveenkumar for their research collaboration.

I would like to thank all my friends and well wishers who have prayed for me during this time. Thanks particularly to Corey, Ritu, Angela, Jebin, Sara, Ingrid, John, Sr. Karen, and Fr. Bryan for being such amazing friends over the last few years.

It is, of course, because of the love and support of my parents that I have reached this far. Thanks also to my brother for taking care of everything while I was away.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

Page

TABLE OF CONTENTS (CONTINUED)

Page

LIST OF TABLES

Table *Page*

LIST OF FIGURES

Figure *Page*

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

LIST OF ABBREVIATIONS

AI	artificial intelligence
BSI	base system intelligence
CA	cluster agent
CCT	controlled covariance tracking
CIQ	control intelligence quotient
DADS	Deployable Autonomous Distributed System
DSN	distributed sensor networks
EKF	Extended Kalman Filter
HIQ	human intelligence quotient
MA	master agent
MIQ	machine intelligence quotient
OOP	object-oriented programming
RMS	root mean square
TT	timed tracking

I. INTRODUCTION

Sensor networks consist of many spatially distributed sensor nodes that can be used to detect and monitor phenomena at different locations. The sensor nodes are usually low-cost, low-power, multifunctional devices having limited processing power, communication capabilities, and battery capacity. Advances in chip manufacturing, sensor, and communication technologies have made it technically feasible and economically viable to develop small sensor nodes that can be used in a variety of application domains. Sensor nodes, also called motes, usually consist of a microcontroller unit, memory, radio for communications, batteries for power, and different sensor elements. Fig. 1 shows the architecture of a typical sensor node. Crossbow, Intel, Moteiv, and Dust Networks are some of the popular mote manufacturers [1-4].

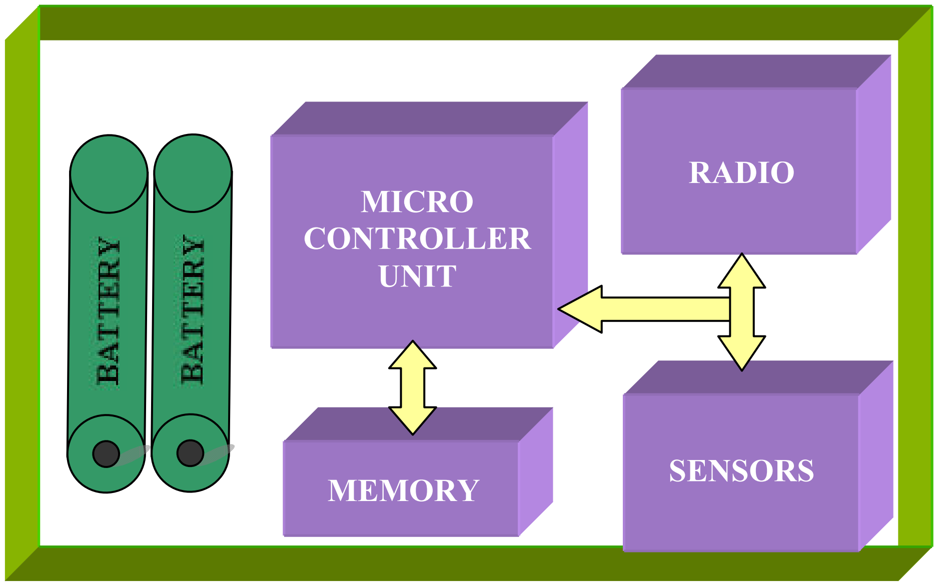


Fig. 1. Sensor node architecture.

Temperature, vibration, position, and light are some of the parameters that can be measured by the sensor node. Applications of sensor networks include habitat and environmental monitoring, health and hospital-related applications, traffic monitoring and management, manufacturing and process control, civil engineering applications, infrastructure security, and military applications, such as surveillance, enemy tracking, and equipment monitoring [5, 6].

Individually, the sensor nodes are limited by their computational and communicational abilities. A large number of nodes that communicate and collaborate with each other can form a sensor network that can overcome these shortcomings. In addition, the small form factor and the inexpensive nature of the sensor nodes allow them to be deployed in large numbers. The sensor network can be formed either over a wired or wireless communication medium. Wireless sensor networks are being increasingly used in a number of application domains since they offer the advantages of reliability, accuracy, cost effectiveness, flexibility, and ease of deployment [5, 6].

In spite of the advantages offered by sensor networks, an individual node's battery charge may be depleted through normal network operation, leaving the node useless and possibly weakening the network. With reduced resources, the network may no longer be able to deliver the required level of performance. The management of power and the other resources that effect field life is an important consideration in sensor networks. The goal of resource management is to conserve the available resources and balance the use of resources in the network.

Agent systems and multi-agent systems are a new and robust paradigm for designing flexible architectures for systems with disparate needs. An agent is an entity that acts in the place of another, with its authority, in order to bring about a desired result. Agents are capable of carrying out goals either alone or as a part of a larger community in which they work together to meet goals. Agents provide a level of abstraction for achieving goals in a system, thereby potentially simplifying the design of a complex system. Agents offer the advantages of lower bandwidth requirements by using processed data rather than raw data for functioning. A higher fault tolerance and better load balancing can be achieved by using multiple agents in the network. Agents can be added, removed, and modified, thus offering design flexibility for distributed sensor systems. Agents are gaining popularity because of their flexibility, modularity, and their general applicability, especially in the field of distributed computing [7].

A. Problem Statement and Approach

While the use of agents for achieving goals in sensor networks has been widely studied, there is a lack of knowledge regarding the comparison of agent paradigms based on the number of agents, agent functionality, and the presence of agents at different levels in the network hierarchy. The primary objective of this dissertation is to present and compare alternative agent paradigms applicable for resource management in sensor networks. The comparisons provide an increased understanding of how the performance of the sensor network changes with the number, functionality, and presence of agents at different levels in the network hierarchy.

A modular simulation framework based on an object-oriented design was developed to facilitate the implementation and comparison of different agent paradigms. This simulation framework was used to generate data for a detailed analysis of component interactions and in performance evaluation. The number of computations, number of communications, tracking performance, and node battery charge remaining for different simulation scenarios in which different agent paradigms are employed were compared. Performance results were compared and contrasted with a measure of machine intelligence of agent-based systems that facilitates comparing alternatives of varying complexities. Comparison results presented in this dissertation may assist designers of agent-based systems in using agents to their best advantage in scenarios similar to those explored here.

B. Contributions

The principle contributions of this research include the following:

- Development of an object-oriented framework for simulation of agent-based distributed sensor networks (DSNs).
- Development of adaptive sampling algorithms for energy-efficient target tracking in DSNs.
- Development of a measure of machine intelligence for comparing agent-based systems.
- Implementation of agent-based resource management algorithms in DSNs.
- Comparisons of agent paradigms for resource management in DSNs.

C. Dissertation Outline

The rest of this dissertation is organized as follows. Chapter II presents a review of DSNs, agents in sensor networks, and some performance metrics that might be useful in comparing agent-based sensor network systems. Chapter III presents the object-oriented simulation framework, the simulation setup, the state space representation of the tracking problem, and the tracking algorithm. Chapter IV presents agent paradigms based on the number of agents and different agent functionalities. Chapter V gives details of the machine intelligence quotient metric developed to compare agent-based systems. Simulation methods describing different agent-based scenarios are presented in chapter VI. Simulation results for different agent-based scenarios are presented and discussed in chapter VII. Finally, chapter VIII presents the conclusions of this research and some areas for future work.

II. LITERATURE REVIEW

This dissertation focuses on agent paradigms for resource management in DSNs. As such, this chapter reviews important and related research in DSNs, agents in sensor networks, and some performance metrics used in system comparisons. The first section covers DSNs and their applications, the challenges faced by DSNs, and some solutions used to overcome these challenges. The second section covers agents and applications of agents in sensor networks. The third section surveys metrics that might be useful in comparing agent-based sensor network systems.

A. Distributed Sensor Networks

DSNs can provide access to information by collecting, analyzing, and processing data. DSNs promise to revolutionize sensing in a wide range of application domains because of their reliability, accuracy, flexibility, cost effectiveness, and ease of deployment [8]. In spite of their advantages and tremendous application scope, sensor networks face some unique challenges due to ad hoc deployment, unattended operation, power constraints, and dynamic changes. An excellent overview of sensor networks is presented in [9, 10]. The applications of sensor networks, the challenges faced by these networks, and some suggested solutions for power efficiency, localization, and routing problems are reviewed in this chapter.

1) Sensor Network Applications:

a) Habitat monitoring: Disturbances caused by human presence while monitoring plants and animals in field environments is a big concern among researchers. A wireless sensor network that eliminated the need for human presence was implanted on an island to measure the usage patterns of nesting burrows and environmental changes in the burrow and surrounding areas [11]. The sensor network could be set up before the onset of the breeding season or other sensitive periods. The sensors could be placed in locations where human access might be unsafe or unwise. The sensor network deployment was expected to be more economical than other alternatives for monitoring over a long period of time.

b) Building risk monitoring: Buildings are at risk for structural degradation, fatigue damage, gas leaks, fires, intrusions, etc. [12]. These risks stem from natural hazards, such as earthquakes, strong winds, and tsunamis, or from man made hazards, such as fire, crime, and terrorism. Therefore, monitoring the various risks in a building is necessary and is an ideal application for sensor networks. These sensors measure accelerations, strain, and displacements; detect fires and intrusions; and apply appropriate risk control measures, such as structural control, maintenance, evacuation guidance, warnings, alarms, fire fighting, rescue, and other security measures. A sensor network consisting of MICA2 motes was demonstrated to be an effective tool for building risk monitoring. A similar application using a wireless sensor network for structural health monitoring was presented in [13].

c) Firebug: The Firebug system uses wireless sensor motes that are deployed in wildfire environments to collect data that allows predictive analysis of evolving fire behavior [14].

d) Deployable autonomous distributed system: Sensor networks have been extensively applied in target detection, classification, and tracking. The Deployable Autonomous Distributed System (DADS) is a network of sensors used for the detection and tracking of ships and submarines [15]. For the DADS project, a large number of sensor nodes will be placed on the ocean floor, near the mouth of a port or bay. These sensors will detect and track ships and or submarines that are moving in and out of the harbor. Depending on the network topology, the sensors will collect data, fuse the data, and then send the data to higher levels in the system. Several algorithms and numerous sensor network architectures have been developed and tested for target detection, tracking, field layout, and reconfiguration [16-20]. Further investigations are being carried out in distributed coordination and control schemes for future distributed sensor network systems by identifying and exploring alternative distributed intelligent system architectures; methods for distributed data fusion, control, and coordination; and strategies that will increase the ability of the field to survive attacks, failures, and accidents.

Other potential applications of underwater wireless sensor networks include seismic monitoring, oceanographic data collection, pollution monitoring, offshore exploration, disaster prevention, assisted navigation, tactical surveillance, and support of underwater vehicles and robots [21, 22].

2) Sensor Networks Challenges and Solutions: In spite of their advantages, the design and development of sensor networks present many difficult challenges. A good summary of the advantages and the challenges of scalability, dynamics, and stringent resource constraints associated with sensor networks was presented in [23]. The need for localized algorithms for overcoming some of these challenges was suggested. Localized algorithms intelligently select necessary nodes for sensing, tracking, and reasoning to avoid flooding the network with useless or redundant data, and are thus able to extend the field life of the sensor network.

A distributed tracking algorithm to predict a target's future position was proposed in [24]. This tracking algorithm was specially aimed to minimize power consumption in the network. The algorithm used a cluster-based architecture for scalability and robustness. The algorithm also provided a distributed mechanism for locally determining optimal sets of sensors for target tracking. The proposed architecture had a robust failure recovery mechanism that could locate lost targets quickly and with low power usage.

An overview of research on cooperative behavior in distributed sensor systems was presented in [25]. In order to achieve cooperative behavior among sensors in the network, the use of a hybrid reasoning architecture based on intelligent agent networks was suggested. The use of an integration agent with the sole purpose of selecting optimal sensors and fusion algorithms for achieving the goals in the system was also suggested.

For the power constraint problem associated with distributed sensor systems, a scheme of censoring sensors for increasing the field life, especially for a hierarchical multi-hop network, was suggested in [26]. By censoring, sensors sent only informative observations to the fusion center and did not transmit those observations deemed uninformative. An efficient power management protocol in which only nodes that were in the vicinity of the target were turned on while all the other nodes were in a power-saving mode was presented in [27].

3) Adaptive Sampling in Sensor Networks: The rate at which data is sampled at each sensor affects the communication and computational resources in a sensor network. Adaptive sampling schemes adjust the data sampling rate intermittently in order to balance system performance and the available sensor network resources.

A Kalman Filter-based estimation technique wherein the sensor used the estimation error to adaptively adjust its sampling rate within a given range was presented in [28]. When the desired sampling rate violated the range, a new sampling rate was requested from the central server. The server allocated new sampling rates under the constraint of available resources such that the estimation error over all the active streaming sensors was minimized.

Backcasting uses adaptive sampling to significantly reduce communications and power consumption while maintaining high accuracy [29]. For this technique, a small subset of the sensors communicate their information to a fusion center, which provides an initial estimate of the environment being sensed, and guides the allocation of additional network resources. The fusion center backcasts information obtained from the initial estimate to the network at large, selectively activating additional sensor nodes in order to achieve a certain target position estimation error.

In staggered sampling, only a small percentage of sensors collect data in each sampling epoch, and simultaneous sampling at all nodes is not required [30]. This approach schedules the sampling of each sensor to enable power conservation by minimizing the amount of data traffic required by each sensor node and by enabling nodes to enter a low-power sleep state. Prediction models were used to schedule the sampling required to maintain the specified data fidelity.

A decentralized adaptive sampling scheme was used for flood monitoring in the FloodNET system [31]. A non-adaptive benchmark was outperformed by an algorithm developed to balance tradeoffs between the need to conserve power and the need to sample in order to gain the information needed to reduce uncertainty.

Target tracking is a more complex application than the monitoring systems described above. Specific control objectives, such as reducing the uncertainty in a target estimate, have been difficult to achieve in multisensor systems where tracking performance must be balanced with the use of system resources. Covariance control methods developed for single-sensor management schemes have been extended to reduce target estimate bias in a multisensor application in which sensor combinations are selected based on the difference between the desired covariance matrix and that of the predicted covariance of each target [32].

B. Agents

The main contribution toward the field of agents comes from the artificial intelligence (AI) community. Ultimately, AI is about building intelligent artifacts, and if these artifacts sense and act in some environment, then they can be considered to be agents. An agent can be defined as an entity that can carry out tasks independently and autonomously in order to achieve certain goals in a system. Development of agent technologies is based on the theories of agency [33]. The theory of agency defines how an agent's information and attitudes are related, how an agent’s cognitive state changes over time, how the environment affects an agent’s cognitive state, and how an agent’s information and attitudes lead it to perform actions. Over the years, numerous definitions of agents have emerged. The term agent is being used so frequently that there is no commonly accepted notion of what constitutes an agent. In this dissertation, an agent is defined as a program that works on behalf of the user, getting inputs from the environment, making decisions, and affecting some change in the output to achieve certain goals. Agents are capable of carrying out goals either alone or as a part of a larger community in which they work together to meet goals. A list of agent properties that may be employed in designing agent-based systems is presented below.

- Some agents perform tasks individually but some work in groups. Agents may be autonomous or they may also cooperate with other agents to carry out more complex tasks than they themselves can handle.
- Some agents are static and some are mobile. Agents may move from one system to another to access remote resources or even to meet other agents.
- Some agents communicate with each other but some do not. Agents have social ability that enables them to communicate with the user, the system, and other agents as required.
- Some agents learn and adapt but some do not. Agents contain some level of intelligence, from fixed rules to learning engines that allow them to adapt to changes in the environment. Agents not only act reactively, but sometimes also proactively.

Agent-based approaches to systems engineering offer several potential advantages. Agent-based systems naturally support the representation of information-driven processes through communication protocols that may closely resemble the way that people communicate. Agents therefore offer the promise of facilitating the very rapid construction of systems with low development and maintenance costs. Agent-based systems also offer high levels of flexibility and robustness in dynamic or unpredictable environments by virtue of their intrinsic autonomy. For example, agents can be provided with abstractions such as objectives and strategies that are not easily supported by classical object models. Agents facilitate knowledge sharing with other agents or objects within the system. Agents allow learning capabilities to be incorporated in a natural way, enabling agent behavior to change with time, based on acquired experience [33].

An agent is characterized by the concepts of situatedness, autonomy, and flexibility. Agents are especially suited for environments that are open and distributed [34]. Agent-based solutions are suited for problems where a classical centralized solution is not appropriate and where the distribution of information and decision-making is necessary [35]. The paper also listed the advantages of agent-based solutions and the barriers that might be encountered while developing practical agent-based applications.

Agent architectures and multi-agent design and simulation environments that enable agent-based multi-satellite systems to fulfill their complex mission objectives have been discussed in [36]. This paper described the different types of agents, the functional breakdown of these agents, different agent skills, and how to use agents in the organization of distributed satellite systems to achieve certain goals. Multi-agent systems consist of several interacting agents and are appropriate for cases where data are distributed and incomplete, individual agents have a limited viewpoint, there is no global control, and computation is asynchronous. Multi-agent systems enhance computational efficiency, reliability, extensibility, maintainability, responsiveness, flexibility, and reuse. With these properties, multi-agent system technology appears to be an ideal candidate to realize autonomous distributed system designs and provides a framework that may be useful in achieving survivability [37].

An excellent overview of software agents was presented in [38]. A topology of agents placing different types of agents into different classes is listed below.

- Collaborative agents – These agents emphasize cooperation with other agents to perform tasks for their owners.
- Interface agents – These agents emphasize autonomy and learning in order to perform tasks for their owners.
- Mobile agents – These agents are computational software processes capable of roaming through the network performing tasks for their owners.
- Information agents – These agents manage, manipulate, or collate information from many distributed sources.
- Reactive agents – Reactive agents do not have an internal model of their environment, instead they respond to the stimulus from the present state of the environment in which they are embedded.
- Hybrid agents – These agents use a combination of any of the above-mentioned agent philosophies within a single agent.
- Heterogeneous agent systems – These systems consists of multiple hybrid agents, each using different agent philosophies.

Some other classifications that ought to be mentioned are competitive agents, proactive agents, and belief-desire-intention agents that can be classified into one or more of the above-mentioned classes of agents. A discussion of the other classifications of agents is beyond the scope of this dissertation.

The hierarchy of an agent-oriented system is shown in Fig. 2. The basic idea is that the environment consists of entities, some of which can be considered as objects. Of the set of objects, some may be agents, and of the set of agents, some may be autonomous agents. The relationships between agents and objects, and the role of object-oriented analysis in multi-agent system development were discussed in [39]. One of the important attributes of agents is that the agents work towards achieving certain predefined goals. Autonomous agents create and pursue their own agenda as opposed to functioning under some other agent. Agents can thus be considered as a subclass of objects, and object-oriented methods can used to develop agent simulations.

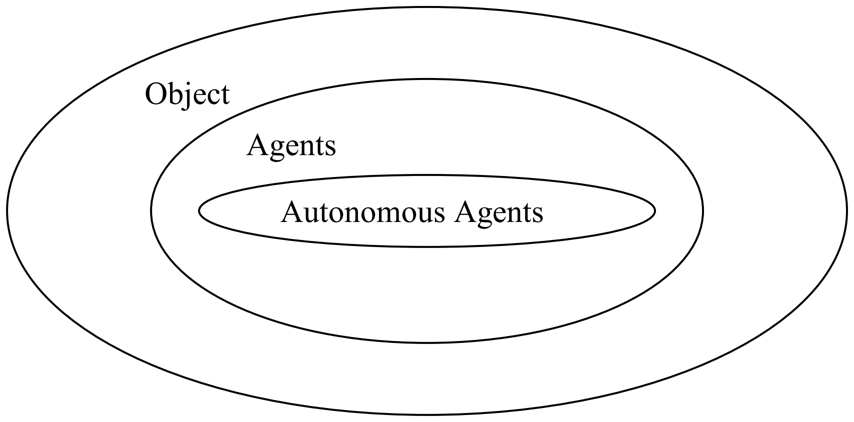


Fig. 2. Hierarchy of an agent-oriented system, showing autonomous agents as a part of the larger group of agents that in turn are a part of the object group.

1) Agents in Sensor Networks: Agents are being used in a wide variety of applications, especially in distributed computer networks systems and distributed sensor network systems. Either hardware or software agents can be used in a network. Some examples are software agents used to filter emails, manage databases, search the internet for data, and control air traffic [40]. Agents have much to offer in sensor networks, and the high interest in bringing together aspects of software agent technology and wireless sensor networks in this emerging research area is indicated by the planning of workshops related to agents in sensor networks as parts of international conferences [41, 42]. The following are representative applications of agents in DSNs.

a) Wildfire tracking: Agilla is a middleware that allows users to create and inject special programs called mobile agents that move around in the wireless sensor networks performing application specific tasks [14, 43]. A detailed case study of Agilla for the Firebug fire-tracking application and a set of application-level performance results that demonstrate the reliability and efficiency of mobile agents in a highly dynamic application were presented in these papers.

b) Urban traffic control system: In a proposed urban traffic control system, adaptive signaling can optimize traffic flow by adjusting the signal based on the current traffic conditions [44]. Intelligent agents provided improved control because of their ability to manage, learn, self-adjust, and respond to non-recurrent and unexpected events. Agents made decisions on how to control an intersection based on goals, perceptions, capabilities, and knowledge of the traffic conditions. The control strategy agent used a prediction of the future traffic state to coordinate with other agents in the system and to plan a signal control strategy.

c) Atmospheric monitoring: An agent-based power-aware sensor network for lower level atmospheric monitoring was presented in [45]. Sensor agents working near the range limit of radio communication kept their radios turned off most of the time to lower power expenditure at the cost of network responsiveness.

d) Other applications: The use of intelligent agents in DSNs and different suggestions on how to use intelligent agents in sensor systems were discussed in [46]. Deployment of mobile agents in DSNs to form an improved infrastructure for multi-sensor data fusion was explored [47]. A target classification example was used to illustrate the efficiency of a mobile agent-based computing model, and it was concluded that mobile agents are a promising solution for high-performance distributed computing [48].

C. Metrics for Comparing Agents-Based Sensor Networks

Existing methods for comparing different software systems can be used for the comparison of different agent-based systems, provided the agents are implemented in software. Hardware agents might use different measures. Uncertainty and complexity are the two important factors affecting the quality of agent-based systems [49]. In multi-agent systems, uncertainty arises when agents are engaged in competitive acts. Complexity can be defined as the degree to which a system has a design or implementation that is difficult to understand. Complexity can be determined by factors, such as the number and intricacy of interfaces, the number and intricacy of conditional branches, the degree of nesting, and the types of data structures. Complexity is structural as well as algorithmic. Structural complexity changes as the system evolves and new components and functions are added or removed from the system. Algorithmic complexity in multi-agent systems arises due to the mechanisms of knowledge processing and sharing as well as the ability to engage with other agents in cooperative, coordinative, and competitive tasks. Some of the measures that might facilitate the comparison of different agent-based configurations are reviewed as follows.

1) Function Point Analysis: The function point metric was devised as a means of measuring software size and productivity [49]. It uses functional logical entities such as inputs, outputs, and inquiries that tend to relate more closely to the functions performed by the software as compared to other measures, such as lines of code. For function point analysis, the system is divided into five components: external inputs, external outputs, external inquiry, internal logical files, and external interface files. Once the system is broken into these components, rankings are assigned to the components, and these rankings are then summed to get the function point count for the system. Function points are becoming widely accepted as the standard metric for measuring software size.

2) Halstead Complexity: The Halstead complexity measure was developed to measure the computational complexity of the program module [49]. This method gives a measure for the algorithmic complexity of the system. The Halstead measures are based on four scalar numbers derived directly from a program's source code, where *n1* is the number of distinct operators, *n2* is the number of distinct operands, *N1* is the total number of operators, and *N2* is the total number of operands. Program length, program vocabulary, volume, difficulty, and effort are five measures derived from these numbers.

3) Cyclomatic Complexity: Cyclomatic complexity is the most widely used member of the class of static software metrics [49]. Cyclomatic complexity may be considered a broad measure of the soundness and confidence of a program. Cyclomatic complexity measures the number of linearly independent paths through a program module. This measure provides a single ordinal number that can be compared to the complexity of other programs. Cyclomatic complexity is often used in conjunction with other software metrics. This measure gives the structural complexity of the system. The cyclomatic complexity number is generally considered to provide a stronger measure of a program's structural complexity than that provided by counting lines of code.

4) Entropic Analysis: Entropic analysis based on the concepts of information theory can be used to measure the complexity of a software program. If the text of the program is considered as a message then the amount of information contained in the program code can be measured. The complexity of a program is inversely proportional to the average information content of its operators. A software complexity metric based on average information content of each operator in the program source code has been developed [50]. In addition, entropic analysis of probabilistic models to express the uncertainty of reasoning, planning, and decision making can be done in order to compare different software systems.

5) Machine Intelligence Quotient: Along with measuring design and code complexity, a measure of intelligence can be useful in comparing intelligent systems, including agent-based systems. However, what may seem intelligent to one person may not seem intelligent to another. One reason for this confusion is that the term intelligence has a broad meaning [51]. Numerous definitions of intelligence and numerous criteria for measuring intelligence have been proposed. One of the simplest defines intelligence as that which produces successful behavior [52]. Intelligence requires the ability to sense the environment, make decisions, and control actions. Advanced forms of intelligence provide the capacity to act successfully under a large variety of circumstances to survive, prosper, and reproduce in a complex and often hostile environment. In guidelines to be followed for developing an intelligence metric, it is asserted that the intelligence metric developed should be application-specific, dynamic and adaptive; reflect generalization capabilities; and be able to evaluate interrelations and interactions among multiple systems [53]. Survivability and competence are mentioned as additional criteria for measuring intelligence [54]. Survivability is the ability of a system to cope with diversity in the environment, as well as internal faults (hardware and software). In a sensor network, the field life can be considered as a measure of survivability. Generally, field life is defined as the time for which the sensor field is able to achieve a desired level of performance. In this dissertation, field life is defined as the simulation duration before which any of the sensor or cluster nodes or all of the master nodes stop working due to depletion of battery charge. Competence is the ability of a system to successfully perform tasks. For a tracking application, tracking performance can be used as a measure of system competence. Expressiveness and perceptiveness have been suggested as important criteria for measuring the intelligence of machines [55]. Expressiveness is a measure of the output richness of an electromechanical system. Perceptiveness is a measure of the fidelity of an electromechanical system's effective mapping from environmental change to output. Entropy, channel sensitivity, environment sensitivity, scenario sensitivity, information sensitivity, and adaptation rate sensitivity are proposed as metrics of autonomy for an intelligent system [56]. Different schemes for measuring the machine intelligent quotient (MIQ) have been proposed, and two of these relevant schemes are discussed as follows.

a) Scheme 1: A formal analysis of the systems software architecture and hardware configurations can be used to measure the intelligence of a system [57]. The *MIQ* is defined as the product of the complexity of the tasks the system can handle (*T*) and the performance in task execution (*E*).

$$MIQ = T \times E \tag{1}$$

where

$$E = w_B B + w_\delta / \delta \tag{2}$$

$$T = w_\gamma \gamma + w_\delta \delta + w_\lambda \lambda + w_\sigma \sigma + w_\kappa \kappa \ . \tag{3}$$

In (2),

$$B = \max_i (\alpha_i + \tau) \tag{4}$$

where B is the behavior response time, w_B is the behavior response time weight, α_i is the node propagation time for node i, and τ is the trigger propagation time. In (3), T is the task complexity ability, γ is the trigger count, δ is the average strand propagation time overall machine, λ is the layering depth, σ is the total strand count in the machine, κ is the total node count in the machine, and $w_\gamma, w_\delta, w_\lambda, w_\sigma, w_\kappa$ are the respective weights that are derived from system analysis.

b) Scheme 2: A systematic approach to measure the MIQ of human machine cooperative systems is presented in [58]. In a human-machine cooperative system *MIQ* is expressed as

$$MIQ = CIQ - HIQ \tag{5}$$

where *CIQ* is the control intelligence quotient and *HIQ* is the human intelligence quotient.

Symbols required for setting up *CIQ* and *HIQ* in (5) are defined as follows. The set of n tasks required to control events is

$$T = \{T_1, T_2, T_3, ..., T_n\}. \tag{6}$$

The intelligence required to execute task T_i is r_i with

$$r = \{r_1, r_2, r_3, ..., r_n\}. \tag{7}$$

The data transfer matrix is

$$\mathrm{F} = \begin{bmatrix} 0\ f_{12} \ \ f_{13} \cdots f_{1n} \\ f_{21}\ 0\ f_{22} \cdots f_{2n} \\ \vdots \qquad \cdots \qquad \vdots \\ f_{n1}\ f_{n2}\ f_{n3} \cdots 0 \end{bmatrix} \tag{8}$$

where f_{ij} is the data quantity transferred from T_i to T_j. Interface complexity c_{hm} is the complexity of transferring data from human to machine, and c_{mh} is the complexity of transferring data from machine to human. The task allocation matrix is A.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} \end{bmatrix} \tag{9}$$

The elements of matrix A can have only binary values 0 or 1. If the machine performs T_i, then $a_{i1} = 1$, and if the human performs T_i, then $a_{i2} = 1$. If T_i can be assigned neither to machine nor human, then $a_{i3} = 1$. Thus, $a_{i1} + a_{i2} + a_{i3} = 1$ for $\forall i, 1 \le i \le n$. Using the symbols defined previously,

$$CIQ = \sum_{i=1}^{n} a_{i1} r_i + \sum_{i=1}^{n} a_{i2} r_i \tag{10}$$

$$HIQ = \sum_{i=1}^{n} a_{i2} r_i + c_{mh} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i1} a_{j2} f_{ij} + c_{hm} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i2} a_{j1} f_{ij} . \tag{11}$$

The equations presented above are for measuring the MIQ of a human-machine cooperative system. This scheme cannot be used directly to measure the MIQ of agent-based systems that do not have a human element. However, this scheme can be adapted and extended to measure the MIQ of agent-based systems, as outlined in chapter V.

D. Summary

A review of literature pertaining to DSNs, agents in sensor networks, and performance metrics used for system comparisons has been presented in this chapter. An underwater DSN system used for tracking ships and submarines is used as a representative application in this dissertation for demonstrating and comparing agent paradigms in sensor networks. Intelligent agents are used for resource management and improving the performance of the sensor network. Though function point analysis, Halstead complexity, cyclomatic complexity, and entropic analysis are good measures for comparing the size, productivity, and complexity of software systems, a measure of the intelligence quotient may be better suited for comparing software systems that have almost the same size and complexity but achieve different goals by virtue of their intelligence. MIQ is a promising metric for comparing scenarios employing different agent paradigms for intelligent goal achievement in sensor networks. In chapter V of this dissertation, the practical and systematic method of measuring the MIQ of human-machine cooperative systems (scheme 2) is adapted and extended to measure the MIQ of agent-based systems.

III. SIMULATION FRAMEWORK

Simulation is a valuable tool for the performance evaluation of the different operational scenarios of a system. Simulations are used to develop and test alternative scenarios, consider tradeoffs, and evaluate the ability of the system to achieve its goals. Simulations provide insight into the parameters that impact the overall performance of sensor network, expose trade-offs, and allow different alternatives to be explored with ease. In this work, a modular system-level simulation was developed for exploring and comparing agent paradigms in DSNs. The first section of this chapter describes the system level simulation for the sensor network system. The second section describes the object-oriented simulation framework. The third section describes the simulation setup, including the target tracking algorithm based on the Extended Kalman Filter (EKF).

A. System-Level Simulation

In this dissertation, an underwater distributed sensor network system used for tracking ships and submarines was used as a representative application for demonstrating and comparing agent paradigms in sensor networks. A pictorial representation of the DSN for this application is shown in Fig. 3. A modular system-level simulation of the underwater DSN was developed [59, 60]. The three-tier hierarchical network is shown in Fig. 4. An activity diagram of the sensor network simulation is shown in Fig. 5. The sensor nodes report the measured range and bearing of the target to cluster nodes that perform local data fusion. The master node then gathers the data from the cluster nodes and performs global data fusion. The master node generates an estimate of the target position and tracks the target through the field. The master node is also responsible for providing information about the target and the network to an external command center. Since the master node is responsible for communicating the target information to the command center, which may be a ship or a satellite, the master node is assumed to have powerful communication capabilities compared to the cluster or sensor nodes.

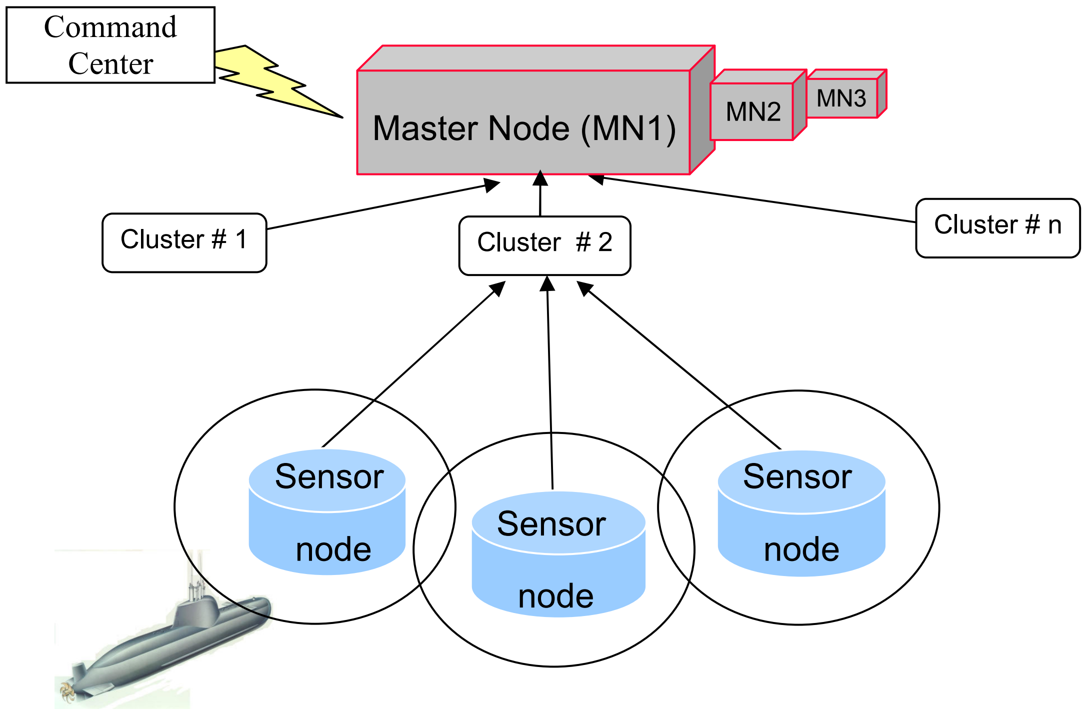


Fig. 3. Pictorial representation of a distributed sensor network (DSN) for detecting ships and submarines. The fusion center gathers data from the clusters and sends the global estimate of the target position to the command center, which may be a ship, satellite, or a naval base.

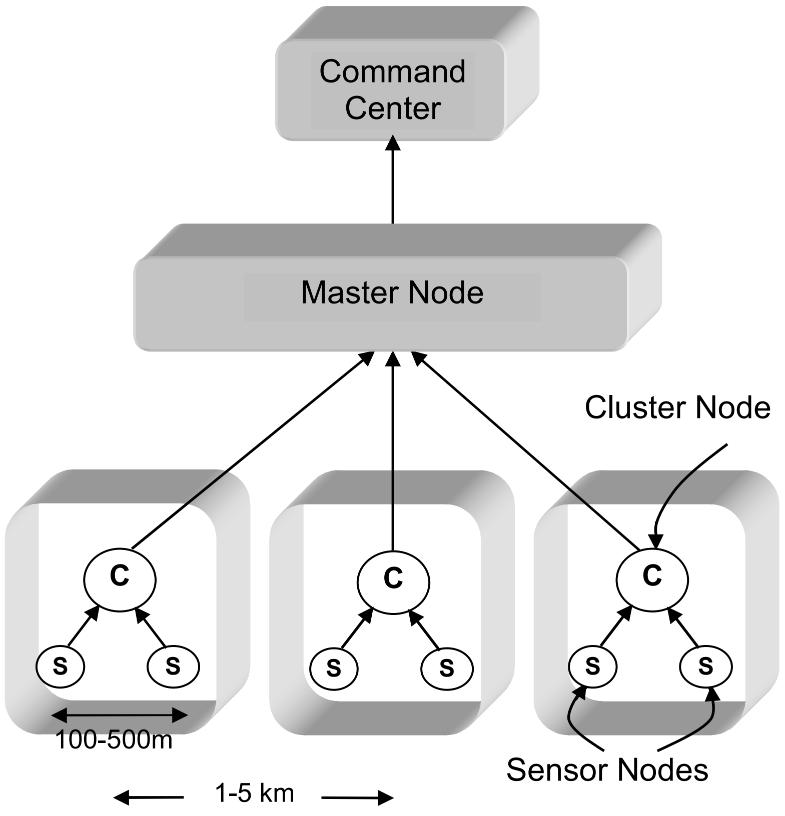


Fig. 4. Three-tier hierarchical sensor network with sensor nodes located on tier one, cluster nodes located on tier two, and the master node located on tier three.

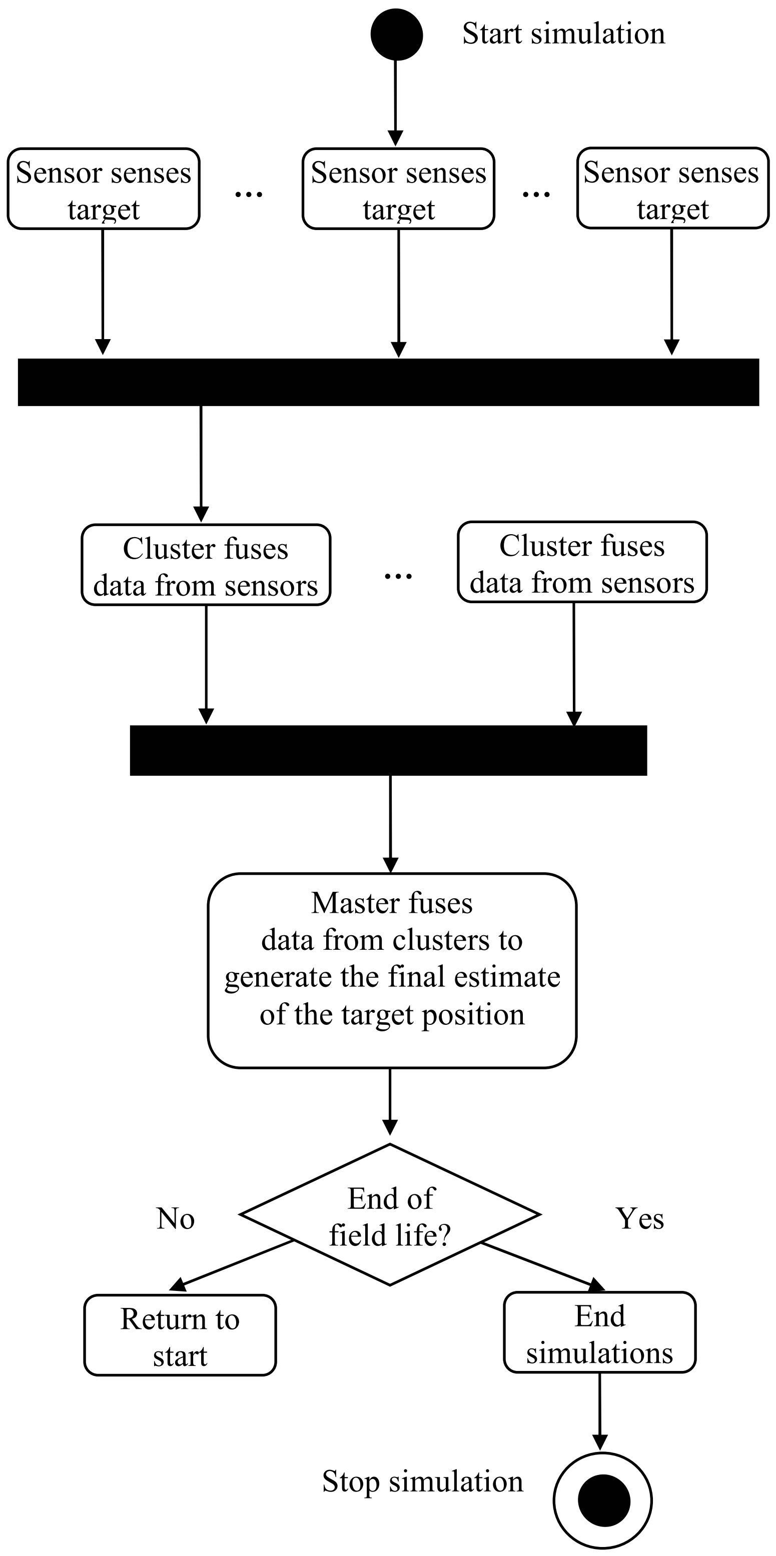


Fig. 5. Flow of activities in the sensor network.

Since the system under consideration is large, having a large number of entities, different strategies, and detailed interactions between the different entities, a modular approach was considered for developing the simulations. Software modules were developed for mimicking the large number of entities and processes that will be present in the actual physical system. The system is partitioned into components that are modeled at a behavioral level appropriate for the goals of the simulation. This modular approach gives the flexibility and extensibility needed to change, add, and remove modules for simulating different scenarios and strategies at a fast pace.

In the system-level simulation, a field layout is initially specified, the target track is generated, and events are simulated at each time step. At every simulation instant, a communications model is run to allow information to be transferred from one node to the other. Next, the node models are run. Fig. 6 shows the operational flow chart of the node model. At the beginning of each time step, a node receives information, such as commands, communication counts, and mode of operation, from the communications module through an inbox. A node then runs a coordination module that coordinates events based on the received information. At the end of the time step, the node sends information to the outbox for transmission. The communication model takes the information from the outbox and sends the message using a communications protocol and appropriate routing to an inbox. Hence, at their fastest, messages placed in the outbox at simulation instant *t* will be available in the receiver inbox at simulation instant *t+1*. However, increased computations at the node, reduced network bandwidth, environmental problems, and queuing at the sensors in the routing path introduce delays in the communication of information to the next level and thereby affect system performance. Communication delays can be simulated inside the communication model in order to allow the affect of delays on tracking performance to be evaluated.

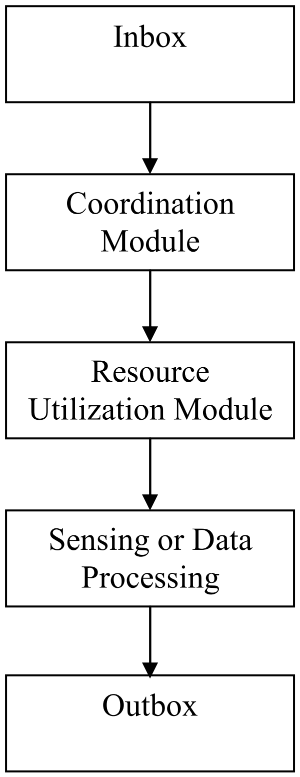


Fig. 6. Operational flow chart of the node model.

Since resource management is one primary focus of this dissertation, the consumption of power resources by each event occurring in the network is measured. Sensing, fusion, communication, and coordination are events that consume power. A count of the number of calls to the communications module and the tracking and control and coordination algorithms is maintained. Proportionate battery-draining weights are assigned to each event depending upon the actual practical node model used for the simulation [59, 60]. Based on these assigned weights, resource utilization is computed at each time step and at each node. To accurately represent resource utilization and battery usage, a battery model is chosen that mimics the charge recovery mechanism as well as the rate capacity effect exhibited by real-world batteries. This phenomenon, called “relaxation,” models how the battery, after discharging at a high rate, regains some of its charge when the discharge current from the battery is reduced or cut off. The relaxation effect gives the battery a chance to recover some of its lost charge, thereby increasing the life of the battery.

B. Agent-Based Object-Oriented Simulation Framework

The system-level simulation described above was modified to accommodate agents in the system. Fig. 7 shows a three-tier, hierarchical, agent-based sensor network architecture. At the lowest level, sensor nodes detect the range and bearing of the target and report this range and bearing information to the cluster nodes. At the second level in the hierarchy, the cluster nodes fuse the data from the sensors and report the local estimate of the target position to the master node. At the third level, the master node fuses data from the cluster nodes to generate the global estimate of the target position. The agents are present on the master node and the cluster nodes. For the distributed sensor network application under consideration, agents present on the master node, called master agents (MA) and agents present on the cluster node, called cluster agents (CA) will be used for target tracking, control and coordination, and reconfiguration of the system and its resources in order to maximize the field life. The agent simulation framework was designed to use the advantages of object-oriented programming (OOP) techniques in system design [61]. MATLAB© OOP techniques were used to develop the simulations.

OOP can significantly increase code reuse and make programs easier to maintain and extend. Fig. 8 shows the object diagram of the agent-based sensor network simulation. The sensor node, cluster node, master node, target, communication, agent, and battery are objects that are modeled for simulations. The objects carry out their functions using the available resources while not violating their predefined attributes. Tables I-VII illustrate the resources and attributes available for the various objects.

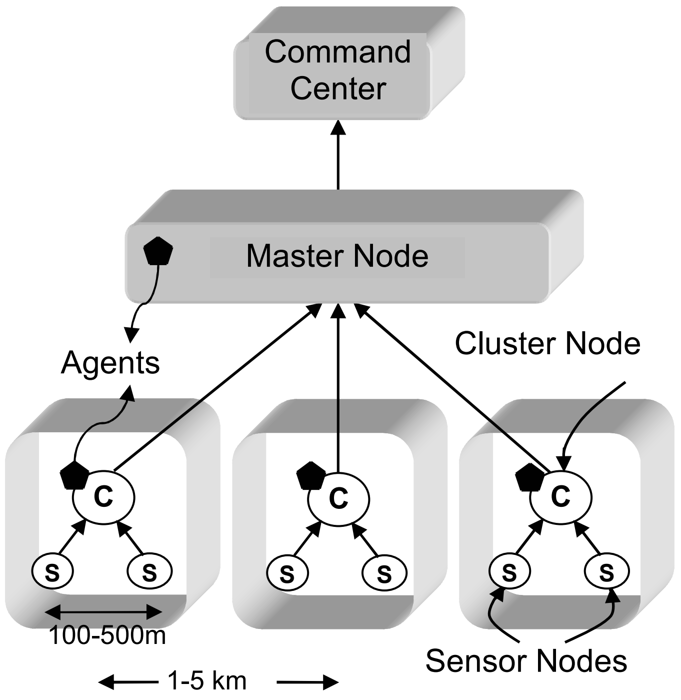


Fig. 7. Three-tier, hierarchical, agent-based sensor network with sensor nodes located on tier one, cluster nodes located on tier two, and the master node located on tier three. Agents are present on the cluster and master nodes.

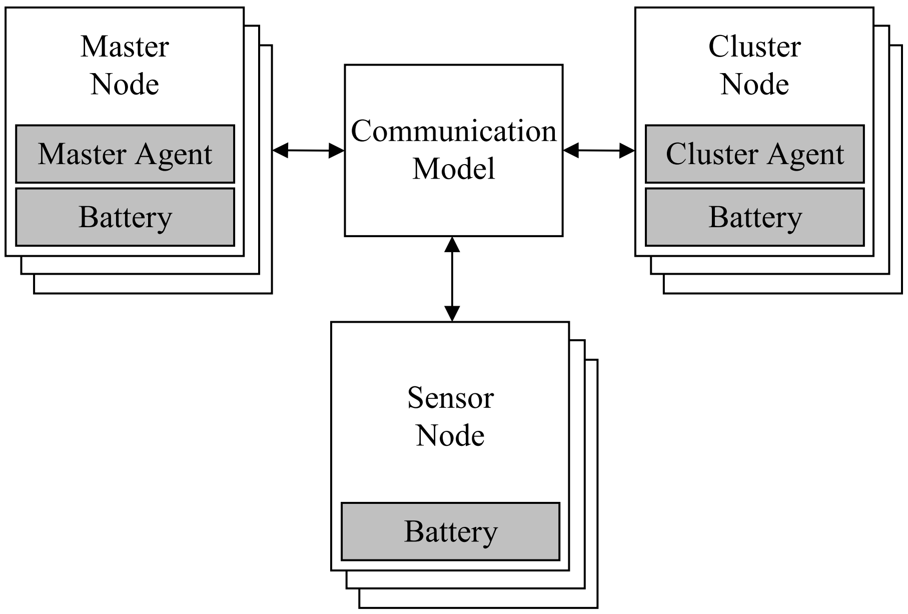


Fig. 8. Object diagram of the simulation framework.

1) Sensor Node: The primary purpose of a sensor node is to detect the target, measure the range and bearing of the target, and relay this information to the respective cluster node. The range and bearing of a target is measured with respect to the local position of the sensor nodes within each cluster. Uncertainty in the knowledge of the sensor positions and the quality of the sensors used in the sensor nodes affect the sensor measurements. The measurement noise variance for the sensor node can be set to accommodate these uncertainties. Sensing and communication weights are set to represent computation and communication resource utilization in the sensor node due to these activities. The sensing range of the sensor node is an adjustable parameter that controls the overlap of sensors in the field. If required, the sensors are also able to route data in the network. Table I presents the resources and corresponding attributes for sensor nodes.

TABLE I
RESOURCES AND CORRESPONDING ATTRIBUTES FOR THE SENSOR NODE

Resources	Attributes
Sensor	Sensor parameters • Detection range • Position (*x, y* coordinates) • Measurement noise variance
Computation	Sensing weights
Communication	Communication weights Communication buffer size Communication range
Battery	Battery parameters (see battery object)

2) Cluster Node: The cluster nodes gather the range and bearing information from their respective sensors, fuse this data to generate a local estimate of the target position, and pass these local estimates to the master node. Data fusion and communication weights are set to represent computation and communication resource utilization in the sensor node due to these activities. Table II presents the resources and corresponding attributes for cluster nodes.

TABLE II
RESOURCES AND CORRESPONDING ATTRIBUTES FOR THE CLUSTER NODE

Resources	Attributes
Cluster	Cluster parameters • Position (*x, y* coordinates) • Number of sensors in each cluster
Tracking algorithm	Process noise Measurement noise
Computation	Data computation weights
Communication	Communication weights Communication buffer size Communication range
Cluster agent	Agent parameters (see agent object)
Battery	Battery parameters (see battery object)

3) Master Node: The master node is present at the highest level in the system hierarchy. The master node gathers local estimates from the cluster nodes and fuses this data to generate a global estimate of the target position. More than one master node can be present in the network, but only one master node is active at any given time in the simulation. Data fusion and communication weights are set to represent computation and communication resource utilization in the sensor node due to these activities. Table III presents the resources and corresponding attributes for master nodes.

TABLE III
RESOURCES AND CORRESPONDING ATTRIBUTES FOR THE MASTER NODE

Resources	Attributes
Master	Master parameters • Position (*x, y* coordinates) • Number of clusters in each master • Active state
Tracking algorithm	Process noise Measurement noise
Computation	Data computation weights
Communication	Communication weights Communication buffer size Communication range
Master agent	Agent parameters (see agent object)
Battery	Battery parameters(see battery object)

4) Target: The target moves through the sensor field. The target can be set up to have random initial positions and velocities or it can have a predetermined set path. The target object is modeled to provide Cartesian coordinates of the target position at any given simulation instant. Table IV presents the resources and corresponding attributes for the target.

TABLE IV
RESOURCES AND CORRESPONDING ATTRIBUTES FOR THE TARGET

Resources	Attributes
Target	Cartesian coordinate parameters • X axis position • Y axis position

5) Communication Model: The communication model is responsible for modeling the network communications. A simple outbox-inbox model is set up to transfer data from the outbox of the transmitting node to the inbox of the receiving node. The communication model takes the information from the outbox and sends the message using a communications protocol and appropriate routing to an inbox. Hence, at the fastest, messages placed in the outbox at simulation instant *t* will be available in the receiver inbox at simulation instant *t+1*. If there are computational or communicational overloads at the nodes, the communication model introduces delays in data transmission to represent these overloads. Table V presents the resources and corresponding attributes for the communications model.

TABLE V
RESOURCES AND CORRESPONDING ATTRIBUTES FOR THE COMMUNICATIONS MODEL

Resources	Attributes
Communication	Communication buffer size
Delay	Delay weights for • Computational overload • Communicational overload

6) Agent: Agents are present on the cluster and master nodes. Agents assist their respective cluster and master nodes in achieving the goals in the network. Fig. 9 shows a simple agent architecture. The agent forms a wrapper around the node and perceives the environment from data collected by the sensors, messages from other agents, or from events that occur within the node. Depending on the goal that has been set for the agent, the agent applies some condition - action rules. These rules help the agent to make a decision on what action has to be taken in order to achieve the goal. For the underwater sensor network target tracking application, the agents assist in data fusion, control and coordination, and reconfiguration of the system and its resources in order to maximize the usage of network resources while maintaining an acceptable tracking performance. Table VI presents the resources and corresponding attributes for the agent.

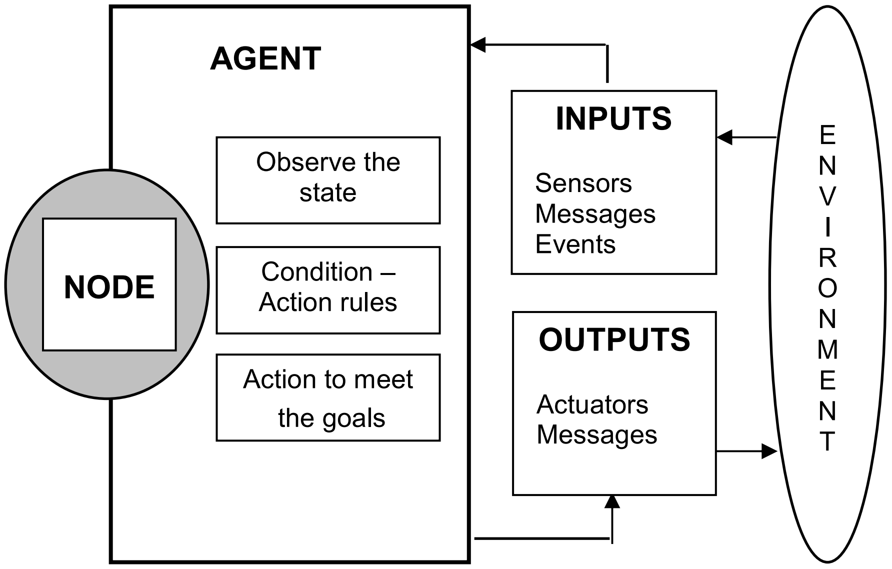


Fig. 9. Simple agent architecture.

TABLE VI
RESOURCES AND CORRESPONDING ATTRIBUTES FOR THE AGENT

Resources	Attributes
Agent	Agent ID Mode (on/off) Agent function

7) Battery: Batteries are present in each node in the sensor network. In this dissertation, battery charge is the term used to represent the life or capacity of the battery. The battery charge is an adjustable parameter that is set according to the simulation requirements. In the node models, the battery weights are set for the active state, sensing, data processing, receiving, and transmitting data. For each of these activities, the batteries are drained proportionately. Table VII presents the resources and corresponding attributes for the battery.

TABLE VII
RESOURCES AND CORRESPONDING ATTRIBUTES FOR THE BATTERY

Resources	Attributes
Battery	Size (life) Battery weight for • Active state • Sensing • Data processing • Receiving • Transmitting

C. Simulation Setup

1) Simulation Flowchart: Fig. 10 shows the simulation flowchart. The various subroutines in the flowchart are described subsequently.

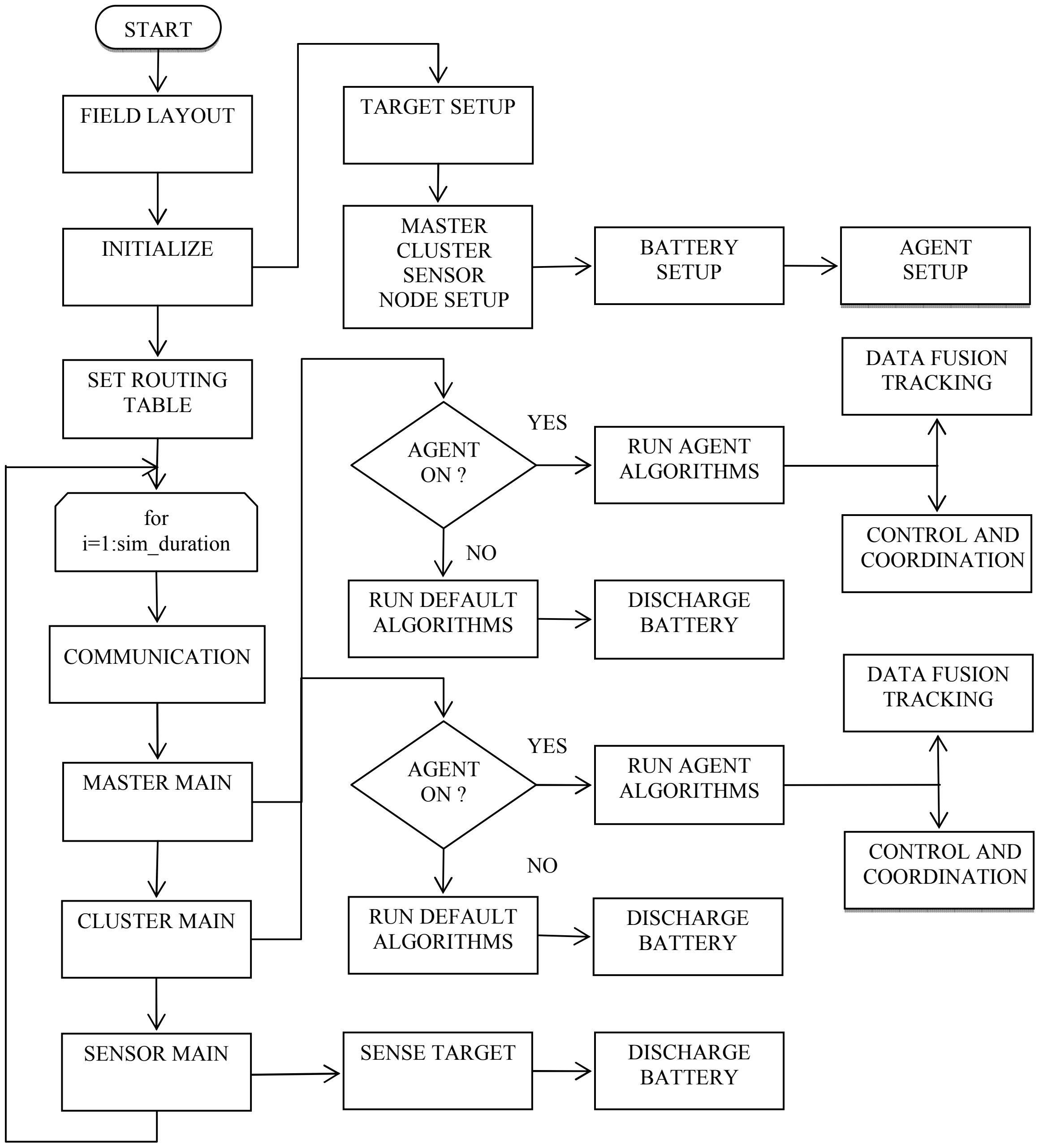


Fig. 10. Flowchart of the simulation.

a) Field layout: The field layout module determines the layout of the nodes in the field. The nodes can be set in the field according to some specialized algorithm or the nodes can have random placements. Fig. 11 shows the field layout of an exemplary sensor network that detects and tracks a target through the field. In this research, the field layout in Fig. 11 is used for simulations, although other field layouts could be easily implemented. The numbers of sensor, cluster, and master nodes are set in the field layout. For simulation purposes, nine clusters and seventeen sensors per cluster are set up in the field. The network has four master nodes (diamonds), nine cluster nodes (dots), and seventeen sensor nodes (small circles) per cluster (Fig. 11). Only one master node is active in the network at any time. The sensor nodes report the measured range and bearing of the target to the cluster nodes that perform local data fusion. The active master node then gathers the data from the cluster nodes, performs global data fusion, generates an estimate of the target position, and tracks the target through the field.

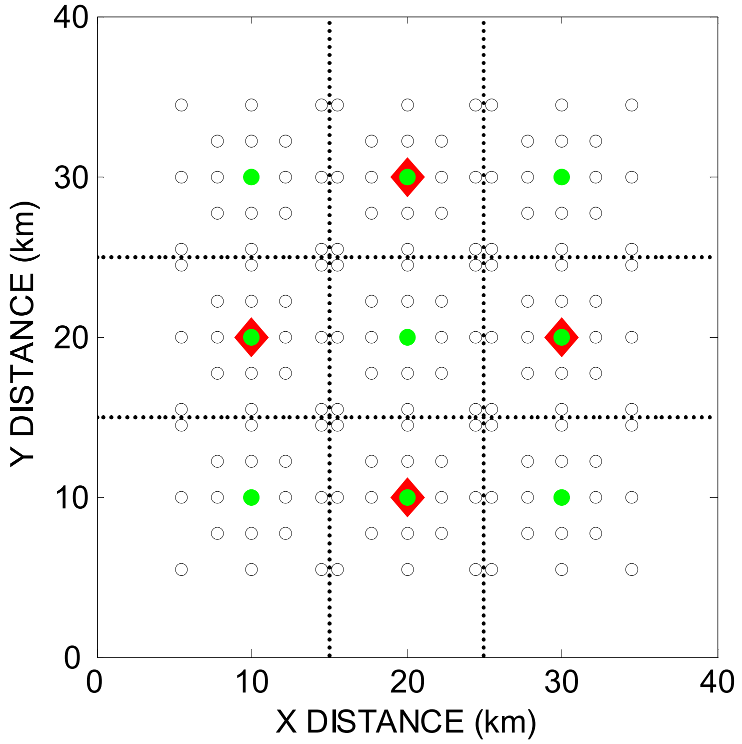


Fig. 11. Field layout of the sensor network. The field includes four master nodes (diamonds), nine cluster nodes (dots), and seventeen sensor nodes per cluster (circles).

Though this field layout (Fig. 11) is not optimal for any specific purpose and is difficult to achieve in an underwater sensor network due to ad hoc deployment, wave action, and ocean floor topography, this field layout (Fig. 11) is useful for simulating and comparing different agent paradigms used for resource management in sensor networks. The sensor node arrangement allows the density of the sensor nodes to decrease from the center to the periphery of the cluster. Changing the sensor density presents challenges to the tracking algorithm. The target position is estimated better when a large number of sensors detect the target simultaneously at the center of the cluster than when the target is detected by fewer sensors at the periphery of the cluster. This field layout (Fig. 11) also enables sensors in adjacent clusters to detect the target when the target is located between clusters.

b) Initialize: The initialize module initializes all the different entities in the network. First, the target track is set up and initialized. Next, the master, cluster, and sensor nodes are set up and then initialized with the different attributes presented in the respective node models. All the nodes are initialized with batteries having a specific charge. The master and cluster nodes are also initialized with the master and cluster agents, respectively.

c) Set routing table: The set routing table module is responsible for computing the routing table of the different nodes in the field according to pre-selected routing protocols. Routing protocols such as fixed multi-hop routing, probabilistic multi-hop routing, modified probabilistic multi-hop routing, and localized optimization routing are available [61, 62]. In localized optimization, the cluster nodes select the sensor node within the cluster that is responsible for routing data to the master node depending on the sensor battery charge remaining. Different routing nodes are thus used within the cluster to achieve uniform battery charge utilization in the sensor nodes. The localized optimization routing protocol is used for simulating the agent-based underwater sensor network system, since it improves the functional field life of the sensor network [61, 62].

d) Communication: The communication module is run for each time step in the simulation. The communication module is responsible for simulating all network activities related to communication. The outbox-inbox communication model described in the system-level simulation section of this chapter is used in the communication module. The communication module transfers data from the sensor nodes to the cluster nodes and from the cluster nodes to the master nodes. The communication module also transfers control signals from the master node to the cluster nodes and from the cluster nodes to the sensor nodes.

e) Master main: The master main module is responsible for simulating all master node activities at each time step in the simulation. The node model (Fig. 6) is used for the master main module. The activities at the master node include processing data from the clusters and making control decisions that are then communicated to the cluster nodes.

f) Cluster main: The cluster main module is responsible for simulating all cluster node activities at each time step in the simulation. The node model (Fig. 6) is used for the cluster main module. The activities at the cluster nodes include processing data from the sensors and making control decisions that are then communicated to the sensor nodes.

g) Sensor main: The sensor main module is responsible for simulating all sensor node activities at each time step in the simulation. The node model (Fig. 6) is used for defining activities performed in the sensor main module. The activities at the sensor nodes include detecting the target, measuring the range and bearing of the target, and communicating this range and bearing information to the cluster nodes.

h) Agent: The agent modules are present on the master node and on the cluster nodes. Depending on the specific goals set for the agent, the agent will have access to specialized data fusion, tracking, control, and coordination algorithms. Based on the input from the nodes, the agents run these algorithms and use condition-action rules to generate the outputs. The agent outputs are sent back to the nodes to specify the control actions that will be taken to control entities in the system.

i) Tracking algorithm: The tracking algorithm is an important module for simulating the agent-based underwater target tracking DSN. Several authors have considered decentralized approaches for data fusion and estimation [63, 64]. In this work, the EKF algorithm, which is well suited for nonlinear multi-sensor systems, is used for data fusion and tracking [65-67]. The EKF algorithm used in the agent-based DSN is described in section 2 of this chapter.

j) Control and coordination algorithms: Intelligent agents can be used to implement control and coordination strategies in order to maximize field life. Agents for control and coordination can gather information about the battery charge remaining in the nodes and modify communication routes through the network. Changing communication routes can help to decrease communication delays in the system as well as to manage the power consumption of the nodes that are being used for communication [61, 62]. Before a node stops working due to depletion of battery charge, the agents can switch functions from the dying node to nodes that still have adequate battery charge remaining [59, 60]. Agents can also be used to selectively turn on and off nodes in the system. Thus, sensor and cluster nodes that are not in the vicinity of the detected target can be temporarily switched off to save battery charge.

2. Extended Kalman Filtering for Distributed Sensor Networks

a) State space representation of the tracking problem: Sensors measure the target range, ρ, and bearing, θ. The velocity of the target is resolved into two vectors in x and y directions, V_x, and V_y, respectively (Fig. 12). In a similar fashion, the range, ρ, of the target from the sensor can be resolved as ρ_x and ρ_y [67].

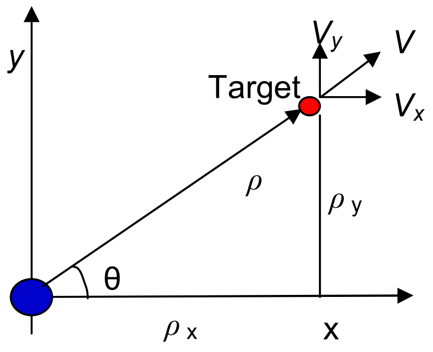


Fig. 12. Tracking problem in two-dimensional space.

If the velocities are assumed to be constant, the state-space model for the target motion is

$$x(k) = Fx(k-1) + w(k) \tag{12}$$

which can also be written as

$$\begin{bmatrix} \rho_x(k) \\ V_x(k) \\ \rho_y(k) \\ V_y(k) \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \rho_x(k-1) \\ V_x(k-1) \\ \rho_y(k-1) \\ V_y(k-1) \end{bmatrix} + w(k) \tag{13}$$

where *T* is the time interval and *w(k)* is the process noise model having process noise covariance

$$Q(k)=E\left[w(k)w^T(k) \right]. \tag{14}$$

The measurement model is

$$\begin{bmatrix} \rho(k) \\ \theta(k) \end{bmatrix} = \begin{bmatrix} \sqrt{\rho_x^2(k)+\rho_y^2(k)} \\ tan^{-1}\left(\dfrac{\rho_x(k)}{\rho_y(k)} \right) \end{bmatrix} + v(k) \tag{15}$$

where *v(k)* is the measurement noise model. The measurement equation is linearized by evaluating the Jacobian of the nonlinear measurement matrix around the predicted state [67]. The linearized measurement update equation is

$$z(k) = Hx(k) + v(k) \tag{16}$$

which can also written as

$$\begin{bmatrix} \rho(k) \\ \theta(k) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ -\sin(\theta)/\rho & 0 & \cos(\theta)/\rho & 0 \end{bmatrix} \begin{bmatrix} \rho_x(k-1) \\ V_x(k-1) \\ \rho_y(k-1) \\ V_y(k-1) \end{bmatrix} + v(k). \tag{17}$$

b) Distributed target tracking: Fig. 13 shows the underwater sensor network architecture used for target tracking. The hierarchical sensor network architecture consists of sensors at the lowest level in the hierarchy. The sensors are arranged in clusters. These sensors measure the range and bearing of the target and pass these local measurements to the cluster node that fuses the measurements to generate the local estimate of the target position. The local estimates ($\tilde{x}, \tilde{P}$) from the cluster nodes are sent to the master node, which fuses these local estimates to generate the global estimate ($\hat{x}, \hat{P}$) of the target position.

The sensor measurements at each cluster are assumed to be obtained synchronously but the measurements and state estimates at the different clusters are assumed to be sequential. The cluster nodes employ a parallel EKF, where all measurements from the sensors are processed at the same time. The time update equation for the local state at the cluster-level is given by,

$$\tilde{x}(k \mid k-1) = F\,\tilde{x}(k-1 \mid k-1) + w(k) \tag{18}$$

where *F(k)* is the state transition matrix, and *w(k)* is the process noise as in (13). The local state covariance prediction is given by

$$\tilde{P}(k \mid k-1) = F\,\tilde{P}(k-1 \mid k-1)F^T + Q(k), \tag{19}$$

where *Q(k)* is the assumed process noise covariance

$$Q(k) = \sigma^2 G G^T \tag{20}$$

with

$$G = \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix}. \tag{21}$$

The *x* and *y* velocities were assumed to be constant in the development of *F* in (13) and in (21). However, this assumption does not reflect the changing velocities observed in practice for target paths as simple as an arc. The choice for *Q(k)* (20) allows tracking of the targets for which the *x* and *y* velocities are not constant. The parameter σ allows tuning of the estimator to accommodate the target paths anticipated for a given application.

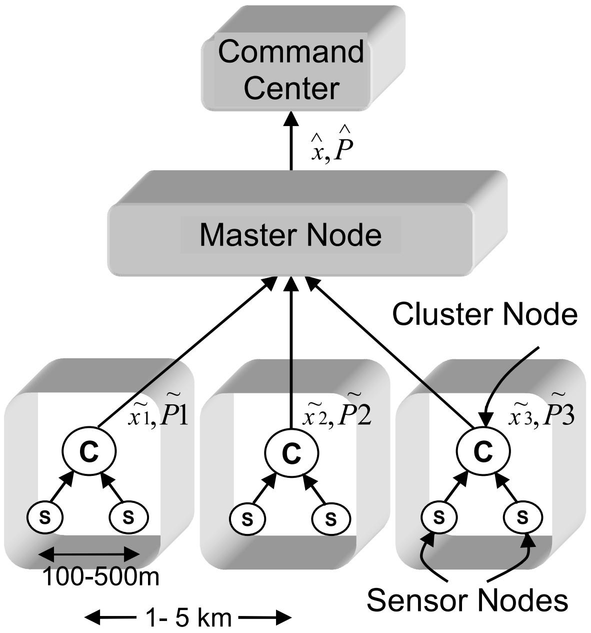


Fig. 13. Three-tier hierarchical sensor network architecture displaying local estimates ($\tilde{x}, \tilde{P}$) at the cluster-level and global estimate ($\hat{x}, \hat{P}$) at the master-level.

The measurement update equation for the local state at the cluster-level is given by

$$\tilde{x}(k \mid k) = \tilde{x}(k \mid k-1) + W(k)[z(k) - H\,\tilde{x}(k \mid k-1)] \tag{22}$$

where *H* is the linearized measurement matrix in (17), *z(k)* is the measurement, and the local state covariance update equation is

$$\tilde{P}(k \mid k) = \tilde{P}(k \mid k-1) - W(k)S(k)W^T(k) \tag{23}$$

with Kalman gain

$$W(k) = \tilde{P}(k \mid k-1)H^T S^{-1}(k) \tag{24}$$

and the innovation covariance matrix

$$S^{-1}(k) = [H\,\tilde{P}(k \mid k-1)H^T + R(k)]^{-1}. \tag{25}$$

The stacked measurement matrix *H* is given by

$$H = (H_1{}^T, H_2{}^T, \ldots, H_n{}^T)^T \tag{26}$$

where *n* is the number of sensors within each cluster having a measurement of the target position. The measurement noise covariance matrix *R* is given by

$$R(k) = E\left[v(k)v^T(k)\right]. \tag{27}$$

The master node employs a sequential version of the EKF (18-27) where the estimates from the cluster nodes are processed one after another [67]. The assumption that the state estimates at the different clusters were time sequential allowed the design of global estimation algorithms to accommodate delayed communication of state estimates from the clusters. For missing state estimates from a cluster, the time update for the global estimate was computed without a measurement update until the cluster estimate became available.

The performance of the EKF algorithm was validated by using the algorithm to track a target moving in a straight line. The expected value of the estimation error achieved for tracking the straight-line target was zero. Also, as expected, the covariance of the global state estimates decreased over time as measurement updates were performed and more information about the target was acquired. The root mean square (RMS) error in the x and y position estimates was closely approximated by the square root of the (1,1) and (3,3) diagonal elements in $P(k)$.

D. Summary

A modular simulation framework based on object-oriented design was presented in this chapter. As discussed in the subsequent chapters, this simulation framework was used to simulate scenarios using different agent paradigms and to generate data for detailed analysis of component interactions and for performance evaluation.

IV. AGENT PARADIGMS

The simulation framework described in the previous chapter was used to develop and explore different agent paradigms. Strategies incorporating single or multiple agents, different agent functionalities, and agents present at different levels in the system hierarchy are presented in this chapter. The first section of this chapter describes agent paradigms based on the number of agents, and the second section describes agent paradigms based on different agent functionalities.

A. Paradigms Based on the Number of Agents

Two agent-based paradigms having single and multiple agents are presented as follows.

1) Single Stationary Agent System: A single agent located on the master node is responsible for target tracking and control and coordination. Centralized single-agent systems are easy to design but possess the disadvantages of centralized systems, including poor survivability.

2) Multiple Stationary Agent System: In multi-agent systems, individual goals may be the same or different for the agents, but the agents work together to achieve the global goals for the system. In a multi-agent system, there may or may not be direct interaction between agents. For a multi-agent system in which the agents interact with each other, the system design becomes complex. In a multi-agent system, agents may be located on the master node and on the cluster nodes. In this work, the master agents are responsible for target tracking, and the control and coordination of the clusters. The cluster agents are responsible for local data fusion and for the control and coordination of the local sensors.

B. Paradigms Based on Agent Functionalities

Different algorithms may be employed by the agents for data fusion, tracking, control, and coordination. The agents have the ability to decide which algorithms to use and when to use them in order to achieve the goal set for the agent. Some of these algorithms are discussed as follows.

1) Agents for Intelligent Data Fusion and Target Tracking: The EKF described in the previous chapter was adapted and encapsulated in the agent to achieve the goals of data fusion and target tracking. Two approaches are described below in which adaptive sampling techniques are used by the agents to balance target tracking performance against resource utilization.

a) Controlled covariance tracking (CCT): In CCT, measurement updates are performed at the master node only when needed to maintain a desired tracking performance. In CCT, the master agent running the EKF tracking algorithm uses the time update to predict the covariance matrix, $P(k+n)$, of the target position estimate n steps ahead. The trace of the master-level $P(k+n)$ represents the sum of the predicted variances of the *x* and *y* position and velocity estimation errors. The trace of the master node *P(k+n)* is compared to a threshold, *Th*, defined as

$$Th = \operatorname{var}\rho_x + \operatorname{var}V_x + \operatorname{var}\rho_y + \operatorname{var}V_y \tag{28}$$

where $\operatorname{var}\rho_x$ is the desired *x* position variance, $\operatorname{var}V_x$ is the desired *x* velocity variance, $\operatorname{var}\rho_y$ is the desired *y* position variance, and $\operatorname{var}V_y$ is the desired *y* velocity variance. Since zero is desired for the *x* and *y* velocity variances, in this work $\operatorname{var}V_x$ and $\operatorname{var}V_y$ are set to zero in (28). If the trace of the master node *P(k+n)* is less than *Th,* then a measurement update is not needed. In this case, the master agent instructs the cluster node to stop sending local estimates to the master node. When no local estimates are available at the master node, only time updates are performed by the master node EKF to obtain the global estimate of the target position. CCT thus reduces the rate at which measurement updates are performed and, as a result, reduces the communication of local estimates from the cluster nodes to the master node while still achieving the desired tracking performance set by *Th*.

Due to the nature of the communication model used in this work, one simulation instant is required for the master agent instructions to reach the cluster node. Another simulation instant is required for the cluster node to act on these instructions and to stop sending local estimates to the master node. Therefore *n* is chosen as 2 when comparing *P(k+n)* to *Th* in order to decide when a measurement update is needed.

For CCT, it is assumed that the performance represented by the threshold can be achieved by sampling at the base EKF sampling rate or slower. Since the measurement update is performed to prevent the variance from exceeding the threshold, the threshold in CCT sets a limit on the expected tracking performance. The RMS range estimation error corresponds to $\sqrt{Th}$. In order to achieve a tracking performance having a low error, *Th* would be smaller than that set to obtain a tracking performance having a larger error; however, with CCT, it should be noted that if *Th* cannot be achieved at the base sampling rate due to infrequent measurements, noisy measurements, fewer sensors detecting the target, or simply because of the nature of the target track, then the tracking performance would be above *Th* even if the measurement updates occur at each and every time step.

The contribution of *Q(k)* in the propagation of the state covariance (19) is important in CCT. The addition of *Q(k)* causes the elements of the covariance matrix *P(k)* to increase at each time step. The diagonal terms in *P(k)* represent the covariance in ρ_x, Vx, ρ_y, and Vy. Therefore, for a given *Q(k)* the threshold selected for CCT implies a maximum time between measurement updates during CCT. Since σ is an important factor in *Q(k)* (20), the choice of the parameter σ for the cluster-level time update affects the tracking of targets with velocities that are not constant, but the choice of σ for the master-level influences the tracking and the maximum time between measurement updates achieved during CCT. In this research, σ is chosen as one in (20).

b) Timed tracking (TT): The timed tracking mode attempts to balance the need to use high sampling rates to track fast-moving targets against the need to avoid depleting a node's battery charge. The measurement update rate is determined within the master node agent using a fuzzy logic algorithm. The structure of the fuzzy logic algorithm is shown in Fig. 14. Between successive measurements, the EKF algorithm performs a time update to predict the target position and velocity. The fuzzy rules set the measurement update rate based on the estimated velocity of the target and the battery charge remaining in the master node that executes the tracking algorithm. The node battery charge is thus conserved using TT. For a node having full or almost full battery reserve, the measurement updates are done less frequently (every 30-35 seconds) when a target is moving slowly through the field (0-10 meters/second) than when the target is moving fast (Fig. 15). Measurement updates are performed less frequently as the battery charge of the node decreases. Therefore, as the simulation duration increases, the battery level of the master node decreases, resulting in a decrease in the measurement update rate, causing a graceful degradation in the target position estimation performance.

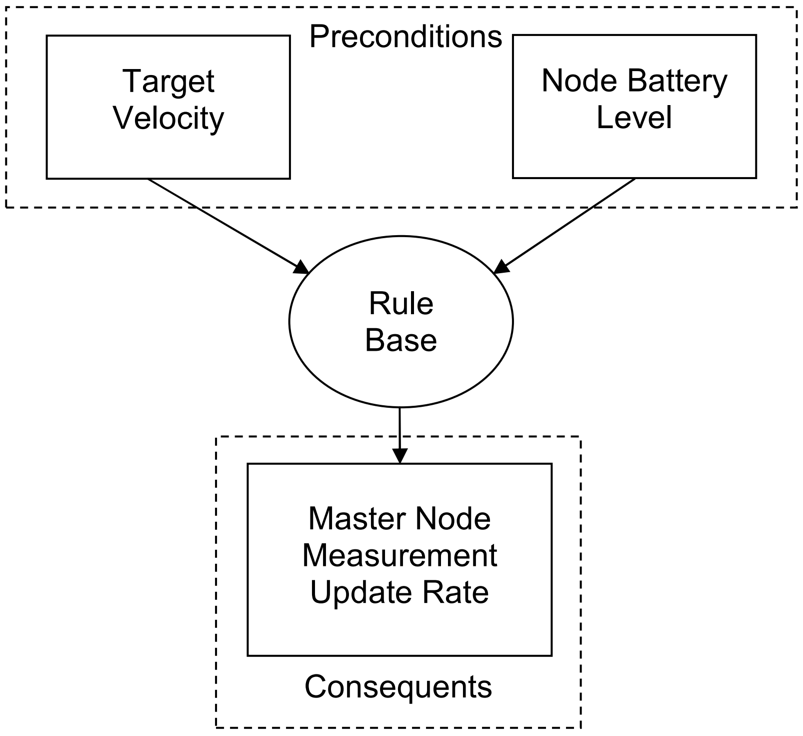


Fig. 14. Structure of the fuzzy logic system for timed tracking (TT).

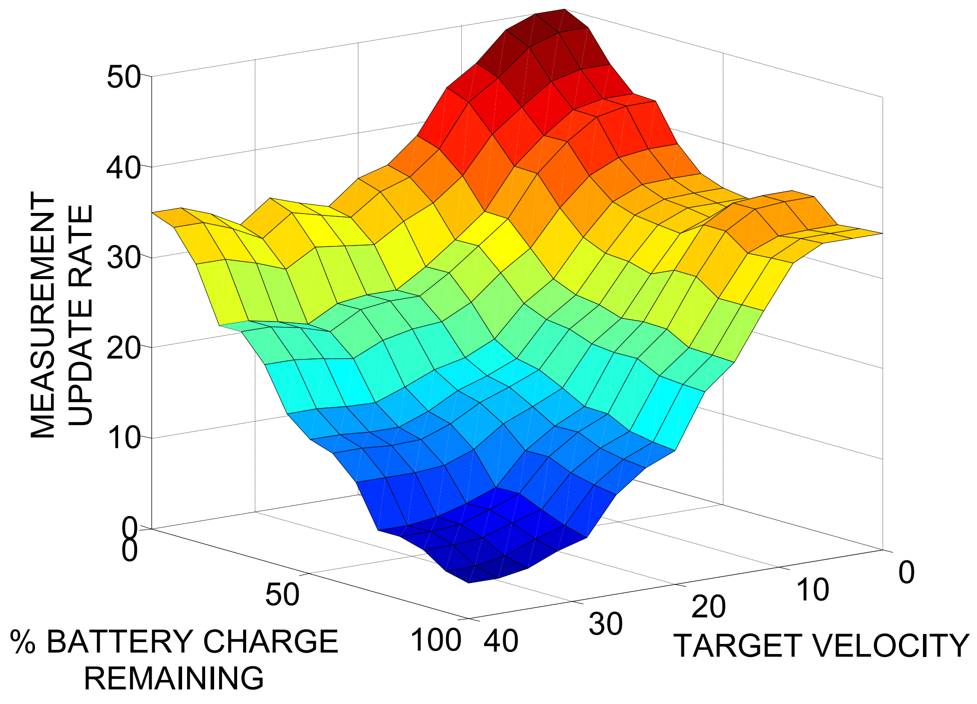


Fig. 15. Surface plot of the timed tracking (TT) fuzzy logic algorithm.

2) Agents for Intelligent Control and Coordination: Different schemes may be implemented by the agents to achieve intelligent control and coordination in order to maximize the useful life of the field. Some of these schemes are described as follows.

a) Control and coordination scheme 1 (CC1): Scheme CC1 is implemented by the master agent when there is more than one master node present in the DSN. In a three-tier architecture, the life of the master node has a significant impact on the useful life of the entire sensor field. The master node is responsible for gathering estimates from the clusters and fusing these estimates to generate the global estimate of the target position. This global estimate is then relayed to the command center. Since the master node has to perform resource-intensive computations and communications, its battery charge is depleted at a fast rate. Loss of the master node represents the end of the useful field life of the network, since without the master node, information about the target cannot be passed onto the command center. Redundant master nodes can be used to overcome this shortcoming [59, 60]. Even though multiple master nodes are present, only one master node and master agent are active at any simulation instant. When the current master node is about to lose its battery charge completely, the master agent at that node transfers the control to a master node in the nearest proximity to the target using scheme CC1.

b) Control and coordination scheme 2 (CC2): Scheme CC2 is implemented by the master agent. In this scheme, the master agent has the ability to instruct the cluster agents to turn the cluster nodes on or off. Only clusters neighboring the cluster where the target is estimated to be present are activated in this scheme [27]. In this research, the clusters are turned on every tenth simulation instant, until a target is detected in the field. Once a target is detected, only the cluster nodes that are at a distance of less than 15 km from the target are turned on, while cluster nodes that are at a distance of more than 15 km from the target are turned off to save battery charge. Fig. 16 shows a representation of how the clusters are activated as the target moves through the field in CC2.

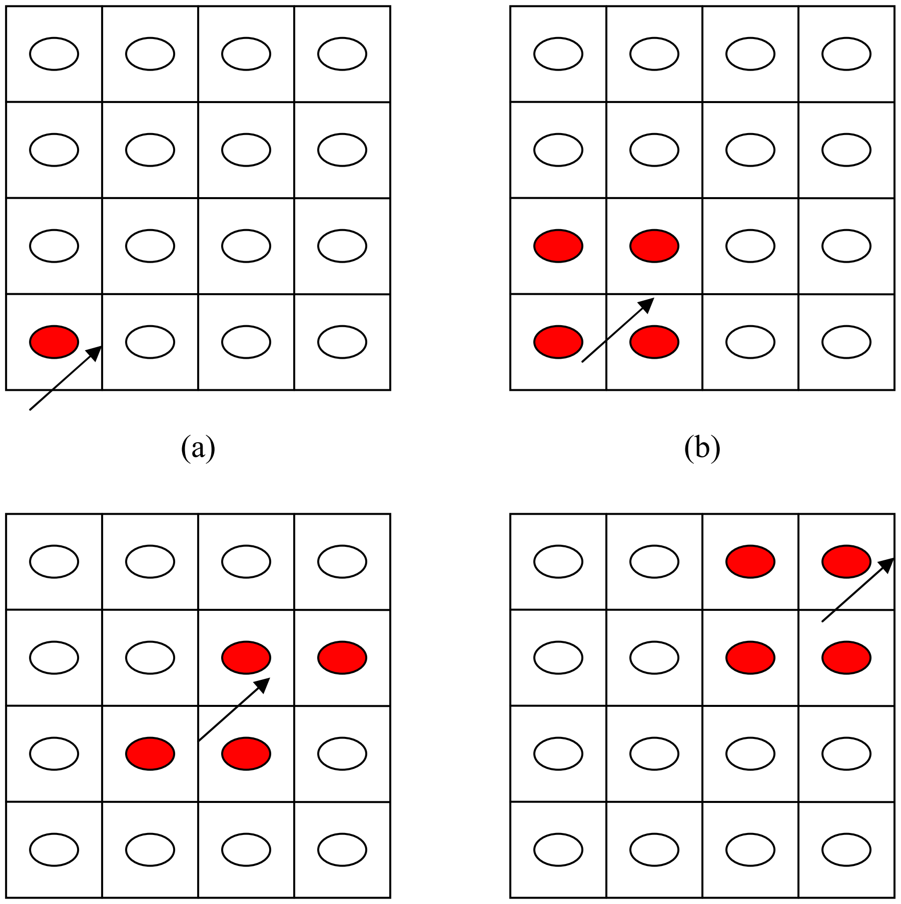


Fig. 16. Activation of clusters as the target moves through the field in CC2. The arrow represents the target as it traverses a path through the field. The solid ellipses are the active cluster nodes, while the remaining clusters shown by transparent ellipses are inactive to conserve battery charge.

c) Control and coordination scheme 3 (CC3): Scheme CC3 is implemented by the cluster agent. When the master agent instructs the cluster agents to turn off the cluster nodes in CC2, the cluster agent in turn instructs the cluster node to turn off all the sensor nodes within its cluster. Only the sensor nodes that act as routing (relay) nodes are kept on, since they are necessary to allow communication with the master node. The cluster nodes using CC2 and their corresponding sensor nodes using CC3 are turned back on when the cluster node distance from the target becomes less than 15 km.

d) Control and coordination scheme 3 (CC4): Scheme CC4 is implemented by the cluster agent. A localized optimization routing algorithm [61, 62] is used for communication in the sensor network. One sensor node within each cluster is used as the routing node. Localized optimization assumes that the cluster agents have knowledge of the remaining battery charge of each sensor node within its cluster. In localized optimization, whenever the remaining battery charge of the routing node falls below a preset threshold (50%), the cluster agent chooses another node within the cluster to act as the new routing node. Localized optimization routing thus helps to balance the power consumption of the sensor nodes used for routing data in the network.

C. Summary

Agent paradigms based on the number of agents and different agent functionalities are presented in this chapter. Adaptive sampling schemes CCT and TT and control and coordination schemes CC1, CC2, CC3, and CC4 used by the agents for resource management were discussed in this chapter.

V. A MACHINE INTELLIGENCE QUOTIENT FOR AGENT-BASED SYSTEMS

The MIQ gives a theoretical measure for quantifying the benefits of the decisions made by agents to achieve goals in the sensor network. The measure of the MIQ of an agent-based DSN is developed by removing the contributions of the human element in (5) and extending the method used for determining the MIQ of a human-machine cooperative system [58].

A. MIQ of Agent-Based Systems

In the sensor network under consideration, agents provide intelligence with the goal of increasing network field life without adversely affecting performance. Even without agents, the system performs certain tasks that are achieved with what will be referred to as base system intelligence (BSI). The agent-based DSN is decomposed into a number of intelligent machines (IMs) working together to achieve system goals, with each IM encompassing an agent (Fig. 17).

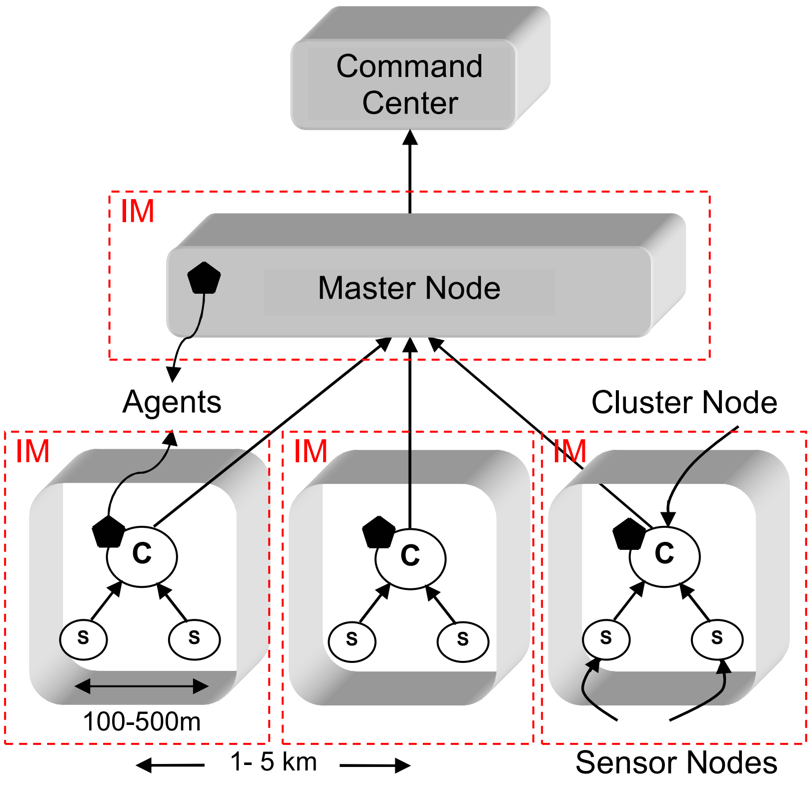


Fig. 17. Agent-based distributed sensor network decomposed into a number of intelligent machines, IMs.

Each IM is assumed to perform eight tasks (detect, observe, identify, interpret, evaluate, define, select actions, and execute actions) in making a decision. The *MIQ* of the total system is calculated as the sum of the *BSI* and *MIQs* of each intelligent machine in the system.

$$MIQ_{total} = BSI + \sum_{ma=1}^{nma} MIQ(ma) + \sum_{ca=1}^{nca} MIQ(ca) \tag{29}$$

where *BSI* is the *MIQ* contributed by the base system, *MIQ(ma)* is the *MIQ* contributed by master agent *ma, MIQ(ca)* is the *MIQ* contributed by cluster agent *ca, nma* is the number of master agents, and *nca* is the number of cluster agents.

The *MIQ* contributed by an agent is calculated using

$$MIQ = \sum_{i=1}^{n} r_i \tag{30}$$

where *n* is the number of tasks performed by each agent (*n*=8) and *r* is the intelligence cost required to perform a task. The task intelligence cost, r_i, for each task performed by different decision-making functions in the IMs is presented in Table VIII. The different decision-making functions that may be implemented by the IM are TT, CCT, CC1, CC2, CC3, and CC4. An IM may implement none or a subset of these decision-making functions. The task intelligence costs were assigned by analyzing the algorithms employed by the agents and the decisions that were made by the agents.

TABLE VIII
TASK INTELLIGENCE COSTS FOR DECISION-MAKING FUNCTIONS PERFORMED BY THE AGENTS

Task	Task No. *i*	No Agents *BSI*	MA TT r_i	MA CCT r_i	MA CC1 r_i	MA CC2 r_i	CA CC3 r_i	CA CC4 r_i
Detect	1	5	3	5	2	5	12	10
Observe	2	3	2	2	2	7	7	7
Identify	3	7	2	2	2	8	8	8
Interpret	4	10	5	8	3	8	8	8
Evaluate	5	0	10	12	6	6	6	6
Define	6	0	5	5	5	5	5	5
Select	7	0	5	5	5	5	5	5
Execute	8	0	10	20	15	18	18	18
Sum =		25	42	59	40	62	69	67

For the scenarios explored in this research, the intelligence required by the IMs to perform two or more decision-making functions is independent. Therefore, the total machine intelligence quotient is given as

$$MIQ_{total} = BSI + \sum_{ma=1}^{nma} \sum_{f=1}^{nf} MIQ(ma, f) + \sum_{ca=1}^{nca} \sum_{f=1}^{nf} MIQ(ca, f) \tag{31}$$

where *f* is one or more decision-making functions performed by the IM. From (29), (30), and (31)

$$MIQ_{total} = BSI + \left\{ \sum_{i=1}^{n} r_i ; \forall f ; \forall ma \right\} + \left\{ \sum_{i=1}^{n} r_i ; \forall f ; \forall ca \right\}. \tag{32}$$

B. Summary

The practical and systematic method of measuring MIQ of human-machine cooperative systems [58] was extended to derive a method for measuring the *MIQ* of agent-based distributed sensor network systems using (32). The MIQ of an agent-based system is a function of the base system intelligence, the number of agents, the number of decision-making functions performed by the agents, and the task intelligence costs required to perform these functions in the system.

VI. SIMULATION METHODS

Scenarios were developed to facilitate the comparison of the different agent paradigms described in the previous chapter using simulations. These scenarios represent combinations of the tracking and control and coordination algorithms implemented by the agents at different levels in the system hierarchy. The simulation framework (Fig. 10) was set up with agents turned on or off as required for each scenario. The network field layout shown in Fig. 11 was used for simulations, and all the nodes in the network were initialized with a battery charge of 100 units.

A. Agent-Based Scenarios

1) Scenario 1: Scenario 1 is the basic setup of the DSN with all the agents turned off. Scenario 1 thus represents the baseline system performing sensing, communications, computations, time updates, measurement updates, and other activities at every simulation instant.

2) Scenario 2: One master node and one master agent are present in scenario 2. The master agent in scenario 2 implements TT. Depending on the measurement update rate set by TT, measurements are transferred from the cluster nodes to the master node; however, the cluster nodes continue receiving measurements from their respective sensors at each simulation instant. Since communication requires considerable battery resources, the number of communications between the master node and the cluster nodes is reduced in scenario 2, thereby potentially saving the battery charge at the respective nodes.

3) Scenario 3: Four master nodes and four master agents are present in scenario 3. The importance of having redundant master nodes in the network is justified in the description of CC1. When the battery of the active master node is about to lose its charge, the master agent at that node transfers control to the master node that is in the nearest proximity to the target (CC1). Since four master nodes are present in the system, scenario 3 can be expected to extend the field life achieved by scenario 1. The cluster agents in scenario 3 also implement scheme CC4, in which the cluster agents perform localized optimization routing in order to balance the power consumption of the sensor nodes used for routing data in the network.

4) Scenario 4: Four master nodes and four master agents are used in scenario 4. The master agents in scenario 4 implement scheme CC1, and the cluster agents implement scheme CC4. The master agents in scenario 4 also implement the adaptive sampling CCT algorithm. A measurement update is performed by the EKF if the trace of the predicted master node, $P(k+n)$, is above Th. Otherwise, only the time update is performed by the EKF. Local estimates are transferred from the cluster nodes to the master node only when a measurement update is required. This intermittent transfer of estimates from the cluster nodes to the master node helps to reduce the number of communications between the master node and the cluster nodes, thereby potentially saving power at the respective nodes. Scenarios 4a, 4b, 4c, and 4d are variations of scenario 4 where the thresholds are set at increasingly higher levels.

5) Scenario 5: Scenario 5 is similar to scenario 4. The master agents in scenario 5 implement schemes CC1 and CCT. The master agents also implement scheme CC2 to turn off clusters that are far away from the target. Cluster agents implement scheme CC3 to turn off sensors that are far away from the target position. The cluster agents also implement scheme CC4 for localized optimization routing. For demonstration purposes, the CCT threshold set for scenario 5 is the same as that set for scenario 4c.

6) Scenario 6: Scenarios 1-5 are employed in a three-tier sensor network architecture having separate physical sensor, cluster, and master nodes. However, the agent abstraction enables the stacking of the master and cluster agent functions on the same piece of hardware. Stacking functional agents helps to eliminate some hardware present in the three-tier sensor network architecture to form a two-tier system. Fig. 18 shows the architecture of the two-tier sensor network with the master agent and the cluster agent stacked on what is essentially a cluster node. Dotted lines represent connections between the cluster node that is not currently acting as a master and other entities in the system. Fig. 19, which is similar to Fig. 11, shows the field layout for scenario 6. The network in scenario 6 has nine cluster nodes (dots) that can also perform the functions of the master node (diamonds).

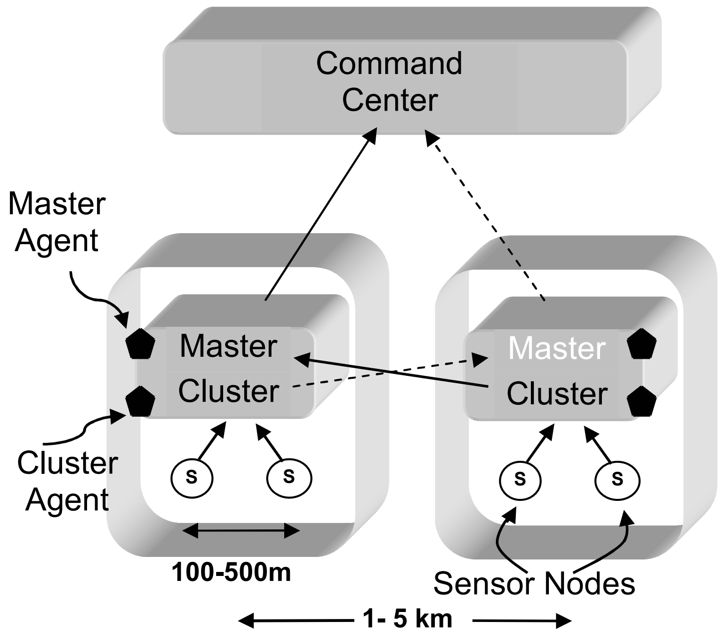


Fig. 18. Two-tier sensor network architecture with stacked master and cluster functional agents on the second tier. Connections with the master/cluster node on which the master agent is currently not active are shown by dotted lines.

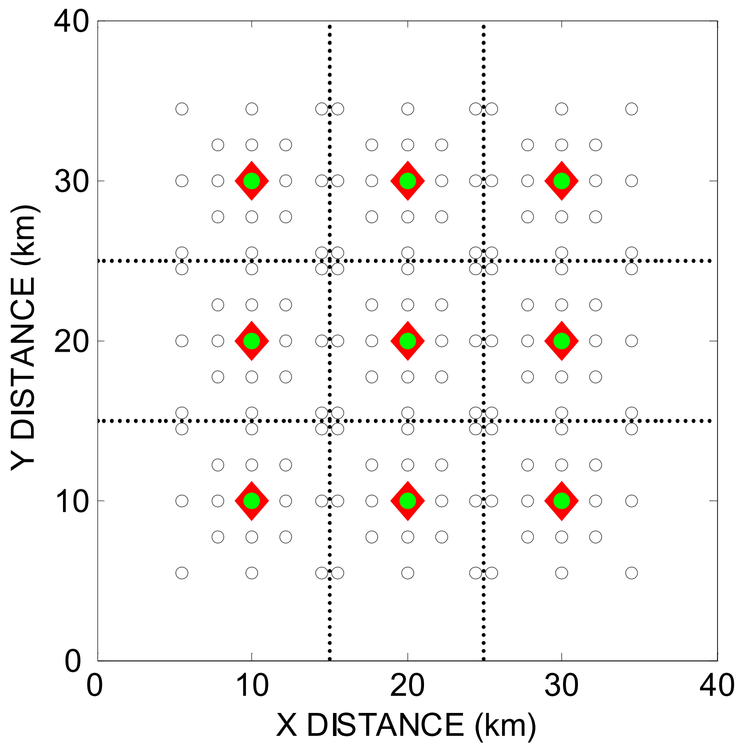


Fig. 19. Field layout of the two-tier sensor network in which cluster nodes also perform the master node functions. There are nine master/cluster nodes (diamond) in the field.

The nodes on which the master and cluster agents are stacked will henceforth be referred to as master/cluster nodes. At any time, one master/cluster node in the network also functions as the master node. Master/cluster nodes that simultaneously run the cluster and master functions have an increased number of computations and communications that deplete battery charge at a fast rate. In order to minimize the battery drain on one cluster alone, the master node functions are transferred from one master/cluster node to another using scheme CC1. If the battery of the master/cluster node having the active master agent falls below 20% of the full capacity, the master node functions are transferred to a master/cluster node in the nearest proximity to the target. This process is repeated until the battery charges of all the master/cluster nodes have fallen below 20% of their capacity. The master agent implements CCT with the threshold set to be the same as in scenario 4c. The master agent also implements scheme CC2 in which only master/cluster nodes adjacent to the cluster where the target is estimated to be present are turned on by the master agent. Cluster agents implement scheme CC3 to turn off sensors that are far away from the target position. The cluster agents also implement scheme CC4 for localized optimization routing.

Table IX summarizes the number of master and cluster nodes, number of master agents (nma), number of cluster agents (nca), and algorithms employed by the agents in different scenarios.

TABLE IX
SCENARIO DESCRIPTIONS

	SCENARIO								
	1	2	3	4a	4b	4c	4d	5	6
Master nodes	1	1	4	4	4	4	4	4	0
Cluster nodes	9	9	9	9	9	9	9	9	9
nma	0	1	4	4	4	4	4	4	9
nca	0	0	9	9	9	9	9	9	9
MA scheme									
TT		X							
CCT				X	X	X	X	X	X
Th				1410	4230	7050	14100	7050	7050
CC1			X	X	X	X	X	X	X
CC2								X	X
CA scheme									
CC3								X	X
CC4			X	X	X	X	X	X	X
Stacked agents									X

B. Target Track

A simple fixed target track was used for single-trial simulations of scenarios 1-6. The target track (Fig. 20) was chosen for scenario comparison, though other complex target tracks could also be easily implemented. A simulation duration of 1100 simulation instants was chosen for the single-trial simulations that allowed the network to track the target and provide a snapshot of the system performance before a large number of nodes started failing. Once nodes start failing in the system, it becomes difficult to compare the performance of agent paradigms in different scenarios. The target track (Fig. 20) was chosen to explore the system tracking performance in different agent-based scenarios while the target moved along a straight-line path, along a curve, along a sharp turn, through areas having different sensor densities, through areas where the target is detected by different clusters, and through areas where the target is detected by multiple clusters. One disadvantage of this target track (Fig. 20) is that the target was confined to only a part of the sensor field. Monte Carlo trials with random target paths were set up to overcome this disadvantage.

The agents in scenarios 5 and 6 employ all the resource management schemes considered in this research, and the best field life performance can be expected from these scenarios. Scenarios 5 and 6 were compared using 40 Monte Carlo trials. Monte Carlo simulation is a method for analyzing and understanding the impact of uncertainty in system. Targets can follow different tracks in the field. However, it is difficult to perform simulations and evaluate the system performance for the entire population of target tracks through the field. The sample mean can be used as an estimate of the population mean for a large (≥ 30) number of samples [68]. Hence, in order to evaluate the mean system performance, a sample (subset) of 40 random target tracks through the field was chosen for Monte Carlo simulations. No constraints were set on the simulation duration for the Monte Carlo trials. For each Monte Carlo trial, a random initial position and velocity was set for the target. The target continued to move with the initial velocity until it was about to exit the sensor field. Whenever the target was about to exit the field, a random velocity was chosen that allowed the target to move while still being constrained to be within the field.

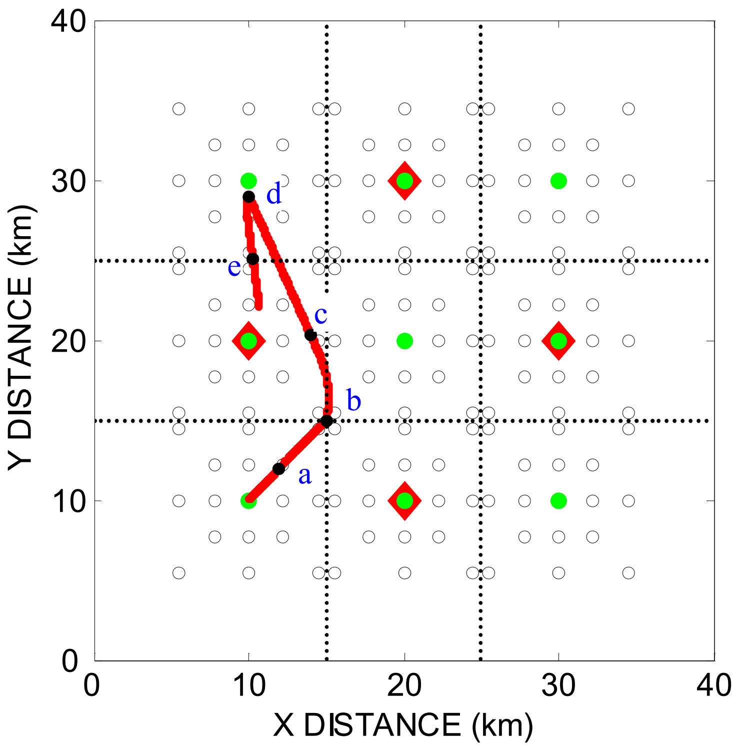


Fig. 20. Target track for single trial simulations. Points a, b, c, d, and e indicate the target positions at simulation instants of 100, 250, 400, 830, and 1000, respectively.

C. Performance Metrics

Performance measurement is important in order to understand the behavior of the sensor network and to ensure that the network is able to meet the design objectives. Field life, tracking performance, number of computations and communications, power reserves, and the MIQ are network-specific performance measures used to compare different agent-based sensor network scenarios.

1) Field Life: Survivability is the ability of a system to cope with diversity in the environment, as well as internal faults (hardware and software). In a sensor network, the field life can be considered to be an indication of survivability. The field life of the network is defined to be the simulation duration elapsed before the battery charge of any sensor, cluster, or master node is depleted. In scenarios having multiple master nodes, the battery charge must be depleted at all master nodes in order for the simulation to end due to loss of the master node function.

2) Tracking Performance: Competence is the ability of a system to perform tasks successfully. For a tracking application, the tracking performance can be used as a measure of system competence. For each simulation run, the error between the actual target position and the estimated target position is calculated. The RMS range estimation error and plots of the covariance, $P(k)$, are used to reflect the tracking performance of the system. These tracking performance measures allow tradeoffs between the field life and tracking performance for the network to be examined.

3) Number of Computations: Data fusion, tracking, control, and coordination algorithms are computationally intensive. A counter accumulates the number of computations assumed to be performed by these algorithms at each node at every simulation instant. Proportionate weights are assigned to the computational algorithms, and the node batteries are drained accordingly.

4) Number of Communications: Communications use battery resources. Information flows from the sensor nodes to the cluster nodes and then on to the master node. The control signals flow from the master node to the cluster nodes and then to the sensor nodes. A counter accumulates the number of communications taking place in the network. Proportionate weights are assigned to the communication tasks, and the node batteries are drained accordingly.

5) Power Reserves: Power reserves indicate the amount of battery charge remaining in the master, cluster, and sensor nodes at the end of simulations. The power reserve is an inverse representation of the power consumption taking place in the node. The number of computations and number of communications performed by the node affect the power reserves for the master and cluster node. The number of sensing activities and number of communications performed by the sensor node affect the power reserves for the sensor node.

6) Machine Intelligence Quotient: The MIQ gives a theoretical measure for quantifying the benefits of the decisions made by agents to achieve the goals in the sensor network. In the sensor network under consideration, agents provide intelligence with a goal of increasing the network field life without adversely affecting performance. The *MIQ* equation (32) presented in chapter V is used to calculate and compare the MIQ of different agent-based scenarios.

D. Summary

Scenarios employing different agent paradigms have been described in this chapter. The target tracks used for single-trial as well as Monte Carlo simulations and the performance metrics used for comparison of different agent-based scenarios have been presented in this chapter.

VII. RESULTS AND DISCUSSION

The scenarios described in chapter VI were simulated using the agent-based object-oriented simulation framework presented in chapter III. Results using a simple target track in single-trial simulations are presented in the first section of this chapter. Results comparing the different agent paradigms employed in scenarios 1-6 are presented in the second section. Discussion of these comparison results is presented in the third section. In the fourth section, results of 40 Monte Carlo trials with random target tracks for scenarios 5 and 6 are presented and compared.

A. Single-Trial Simulation Results

Single-trial simulations were conducted for scenarios 1-6. The simulation duration and the simple target track used for single trial simulations were adequate to capture the behavior of different agent paradigms in the sensor network.

1) Scenario 1: Figs. 21-23 show the simulation results for target tracking in scenario 1. In scenario 1, only one master node was present. Since all the nodes in the system were performing activities at every simulation instant and power was used at the maximum rate, the simulation stopped at simulation instant 517 (Fig. 22) due to the loss of the only master node in the system (Fig. 21c). Battery charge was depleted rapidly in the sensor nodes within the clusters that were used for relaying data to the master node (circled region in Fig. 21a). As the target curved along its path (Fig. 20) between points b and c that represent simulation instants 250 and 400, respectively, the sensor network was unable to follow the target, resulting in large bearing estimation errors between simulation instants 250 and 400 (Fig. 22). Due to the overlapping sensor ranges, multiple sensors were able to detect targets that were close to the center of a cluster. However, as the target moved away from the center of the cluster, fewer sensors were able to detect the target. Fewer sensors detecting the target increased the uncertainty of estimation at the cluster-level, which in turn was reflected by an increase in the trace of the master node covariance matrix, *P(k)* (Fig. 23). The master node *P(k)* indicates the uncertainty of estimation at the master-level; therefore, the trace of the master node *P(k)* (Fig. 23) was higher when the target was at the periphery of the clusters (Fig. 20) than when the target was near the center of a cluster.

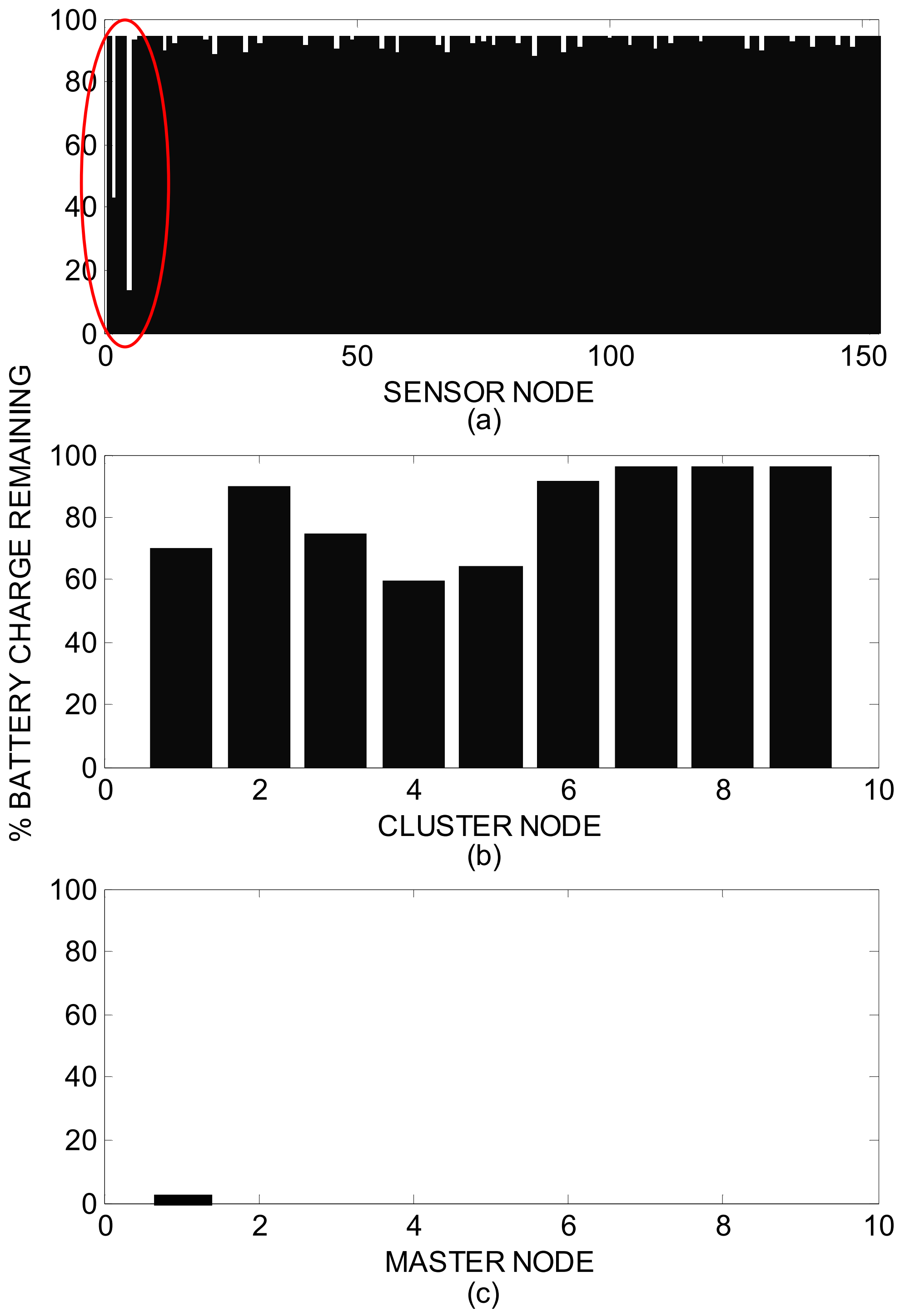
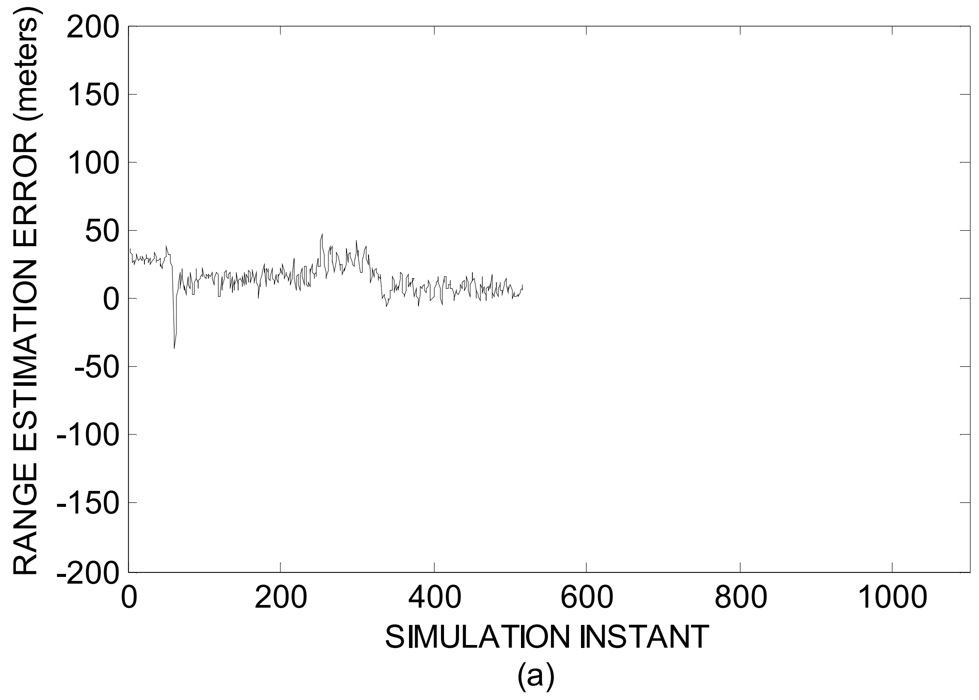


Fig. 21. Percentage battery charge remaining at the end of 517 simulation instants in the sensor nodes (a), cluster nodes (b), and the single master node (c) for scenario 1. The simulation in scenario 1 stopped at simulation instant 517 due to the loss of the only master node in the network. The circled region in (a) indicates the battery charge remaining in sensors nodes in different clusters that were mainly used for routing data in the network.



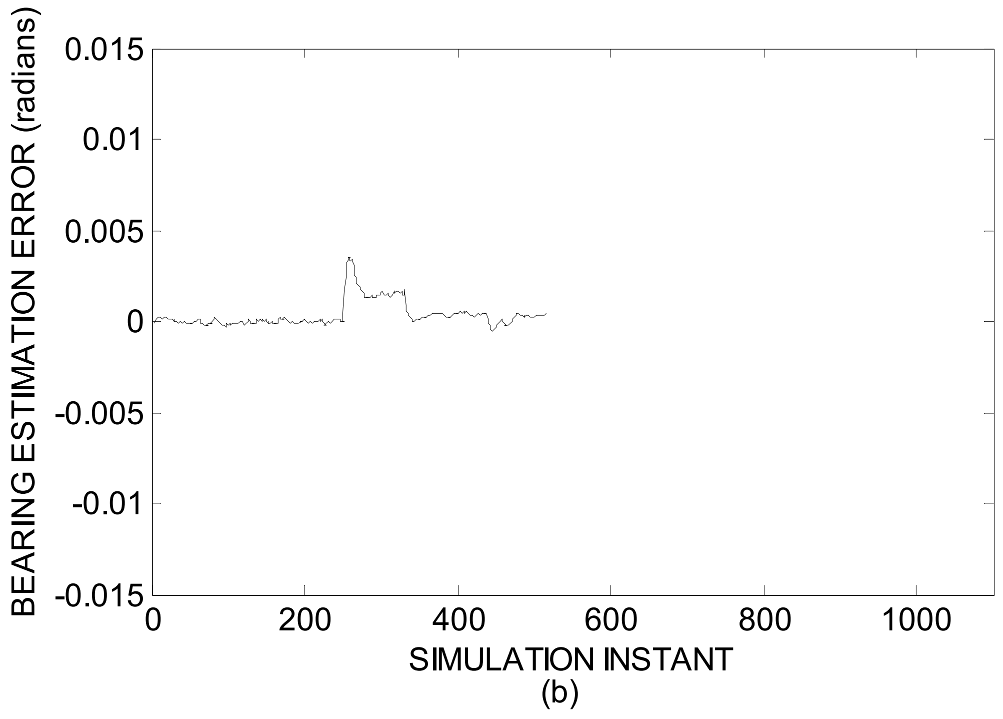


Fig. 22. Range estimation error in meters (a) and bearing estimation error in radians (b) for scenario 1. As the target curved along its path between points b and c (Fig. 20), the network in scenario 1 was unable to follow the target, resulting in large bearing estimation errors between simulation instants 250 and 400. The simulation in scenario 1 stopped at simulation instant 517 due to loss of the only master node in the network.

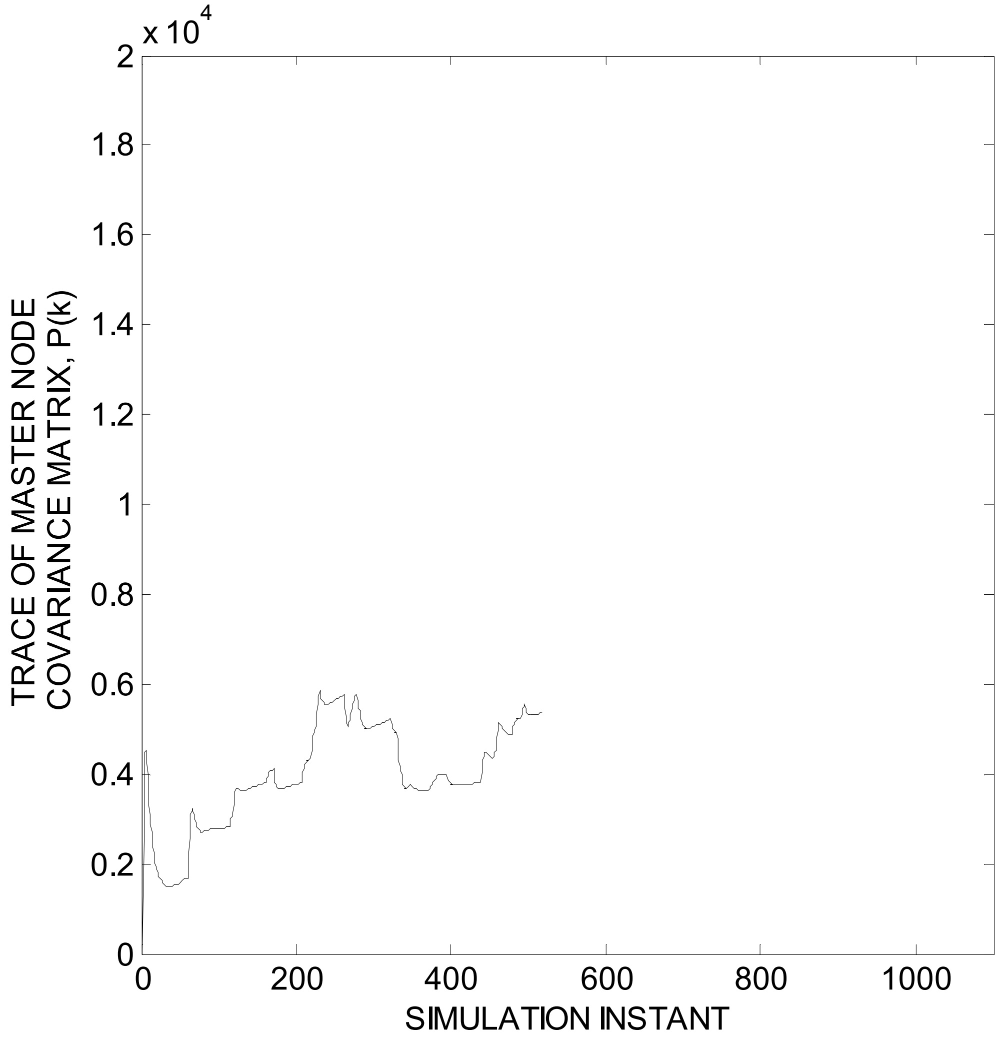


Fig. 23. Trace of the master node *P(k)* for scenario 1. The simulation in scenario 1 stopped at simulation instant 517 due to loss of the only master node in the network.

2) Scenario 2: Figs. 24-26 show the simulation results for target tracking in scenario 2. In scenario 2, only one master node was present in the network. The network in scenario 2 was able to track the target for the entire duration of 1100 simulation instants (Fig. 25). TT was implemented in scenario 2 to adjust the rate at which clusters sent their local estimates to the master node. The master node performed measurement updates only when it received data from the cluster nodes. As the master node battery charge decreased, TT reduced the rate at which measurement updates were performed at the master node, resulting in increased tracking errors as the simulation progressed (Fig. 25). Fewer measurement updates resulted in a large uncertainty of estimation shown by an increase in the trace of the master node *P(k)* (Fig. 26). TT helped to reduce the number of communications between the cluster nodes and the master node. As the target curved along its path (Fig. 20) between points b and c, and d and e, which represent simulation instants 250, 400, 830, and 1000, respectively, the network was unable to follow the target, resulting in large bearing estimation errors between these simulation instants (Fig. 25b). Reduced communications between the master and cluster nodes helped to conserve battery charge at the respective nodes (Fig. 24) and resulted in less use of the routing sensor nodes in the system (circled region in Fig. 24a).

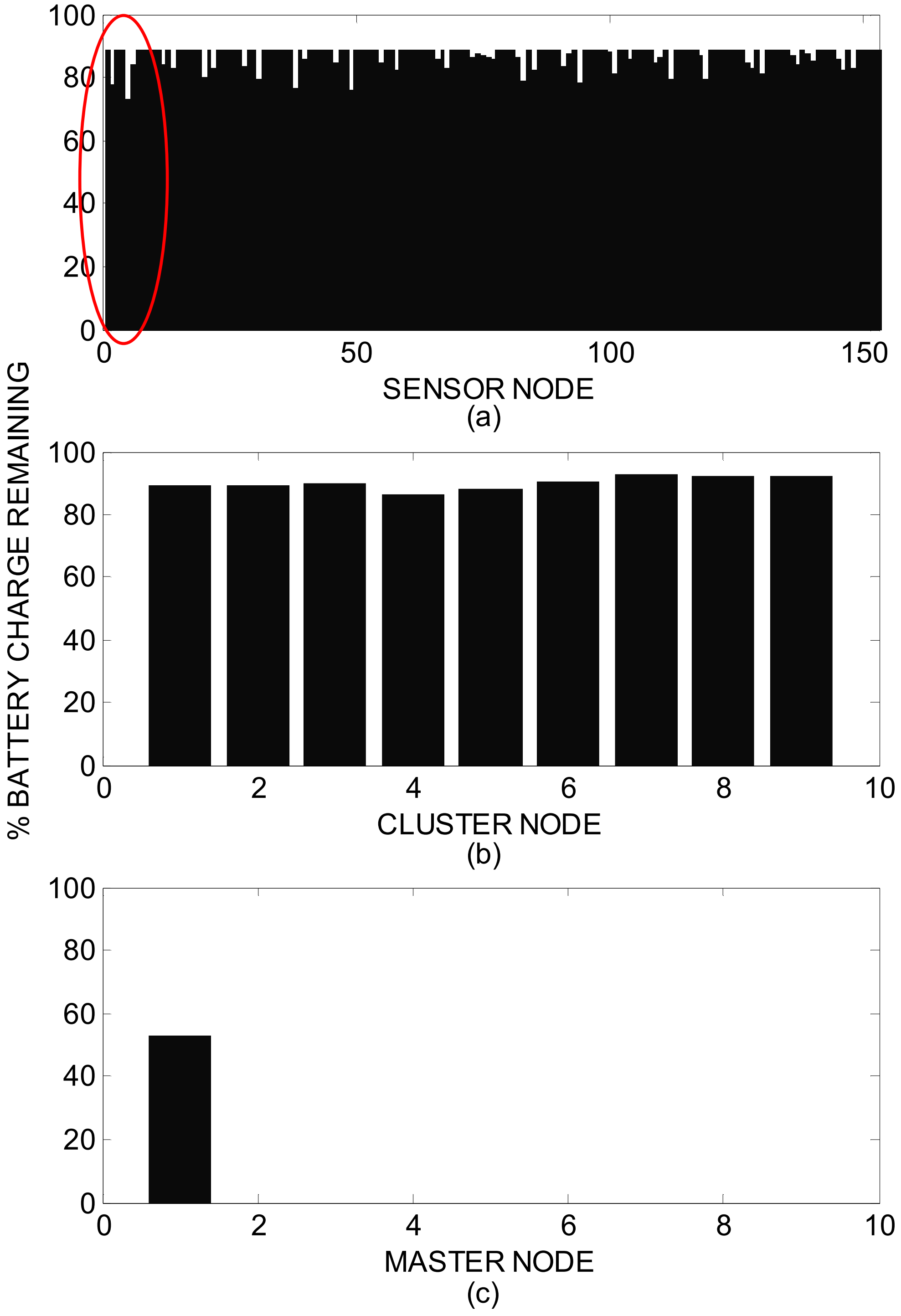
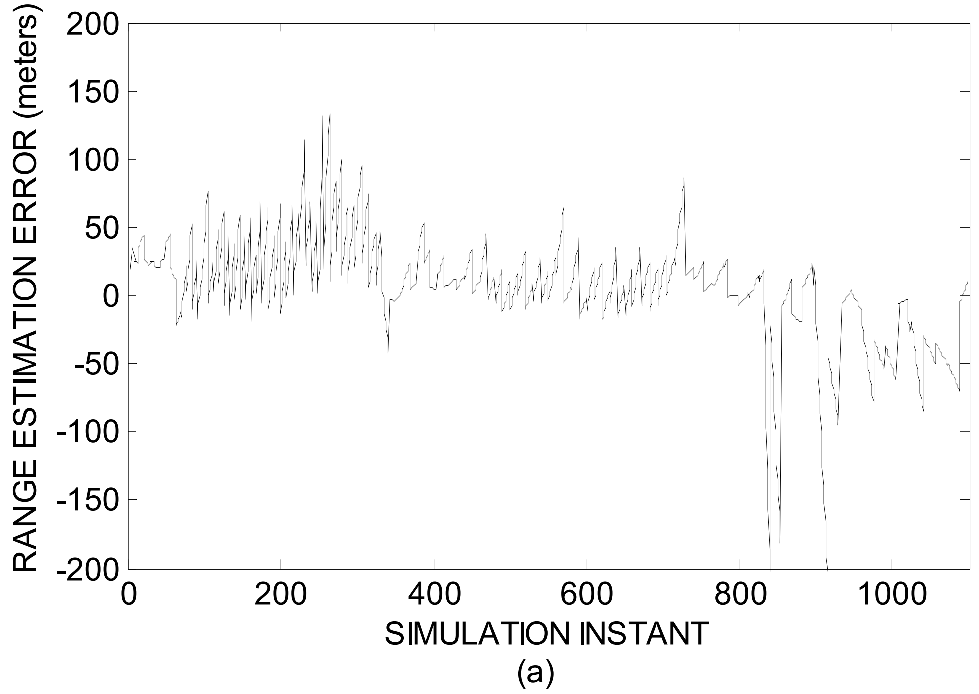


Fig. 24. Percentage battery charge remaining at the end of 1100 simulation instants in the sensor nodes (a), cluster nodes (b), and the single master node (c) for scenario 2.



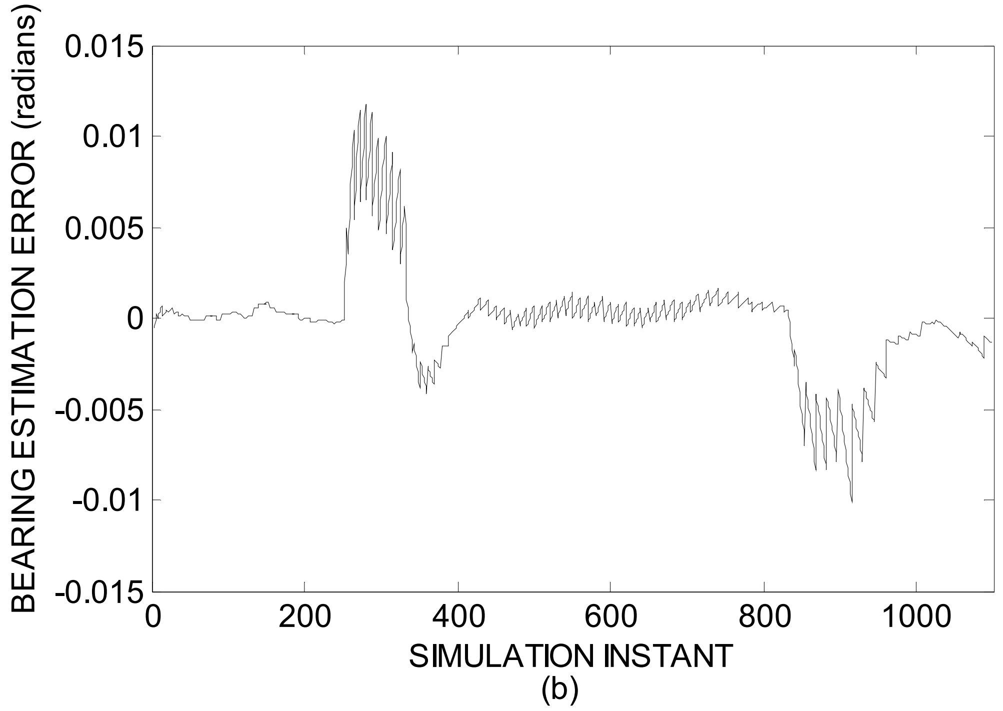


Fig. 25. Range estimation error in meters (a) and bearing estimation error in radians (b) for scenario 2. As the simulation duration progressed, the range estimation errors increased due to infrequent measurement updates at the master node. Bearing estimation errors were exhibited when the target curved through the field between simulation instants 250, 400, 830, and 1000.

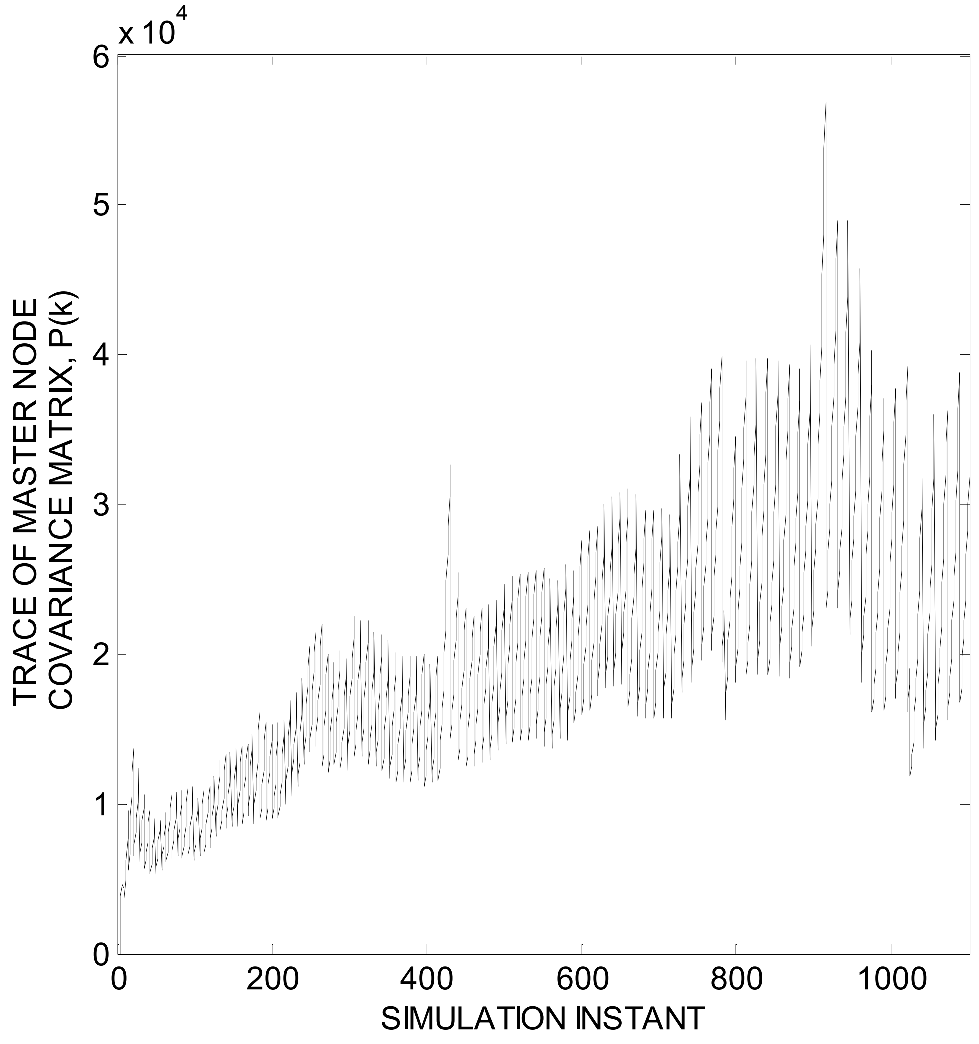


Fig. 26. Trace of the master node *P(k)* for scenario 2. As the simulation progressed, fewer measurement updates performed by the master node resulted in large values for the uncertainty of estimation.

3) Scenario 3: Figs. 27-29 show the simulation results for target tracking in scenario 3. In scenario 3, four master nodes were present in the system, but only one was active at any time. Scheme CC1 was used to rotate the master node functions among the redundant master nodes; thus, the sensor network was able to track the target for the entire simulation duration of 1100 simulation instants (Fig. 28). Since the system was tracking the target at every instant, the range and bearing estimation errors achieved in scenario 3 were small (Fig. 28), except when the target was moving along a curve. Bearing estimation errors were higher as the target curved along its path (Fig. 20) between points b and c, and d and e, which represent simulation instants 250, 400, 830, and 1000, respectively (Fig. 28b). The cluster agents in scenario 3 also implemented scheme CC4 for localized optimization routing. In CC4, when the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster node rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes (circled region in Fig. 27a).

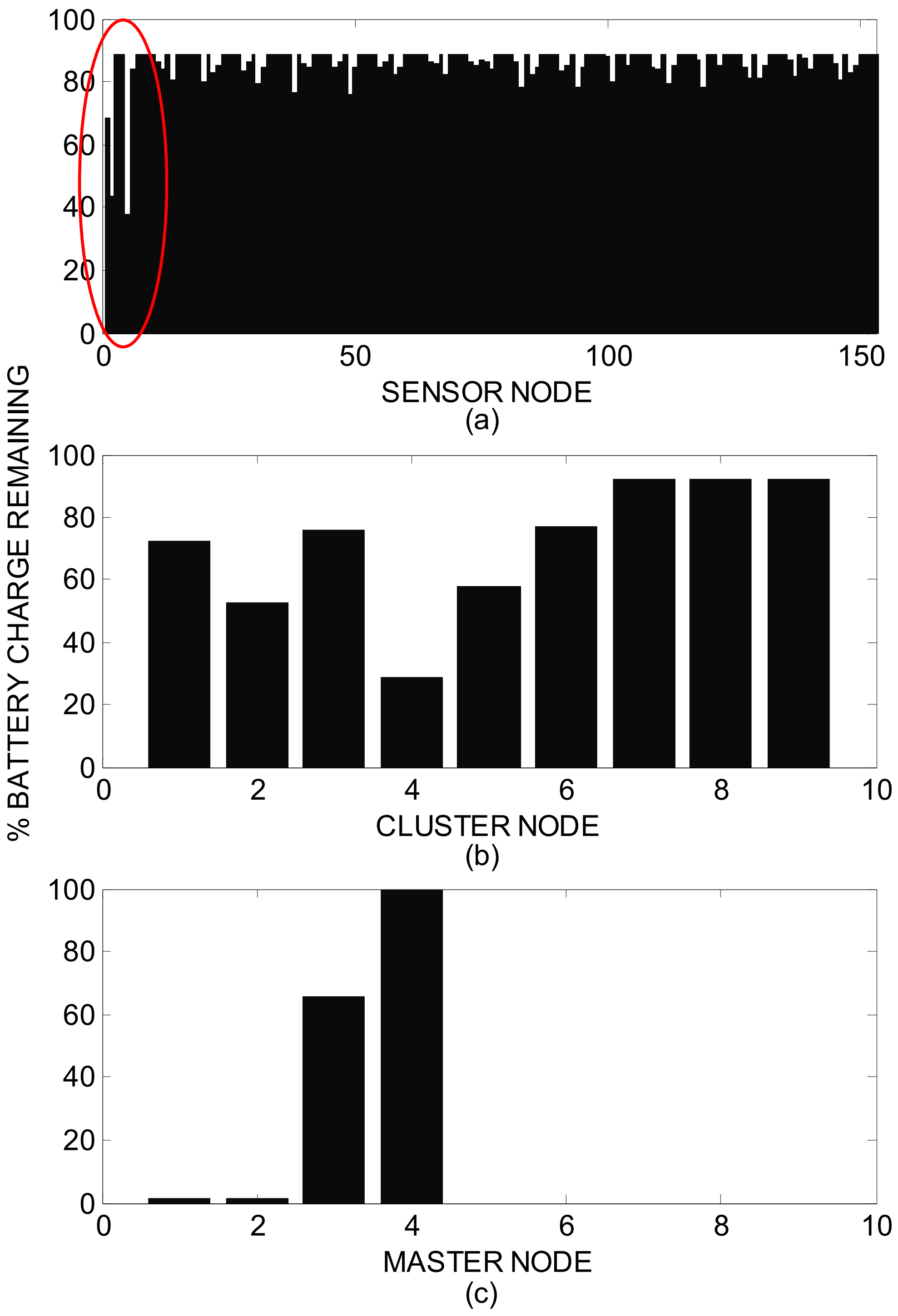
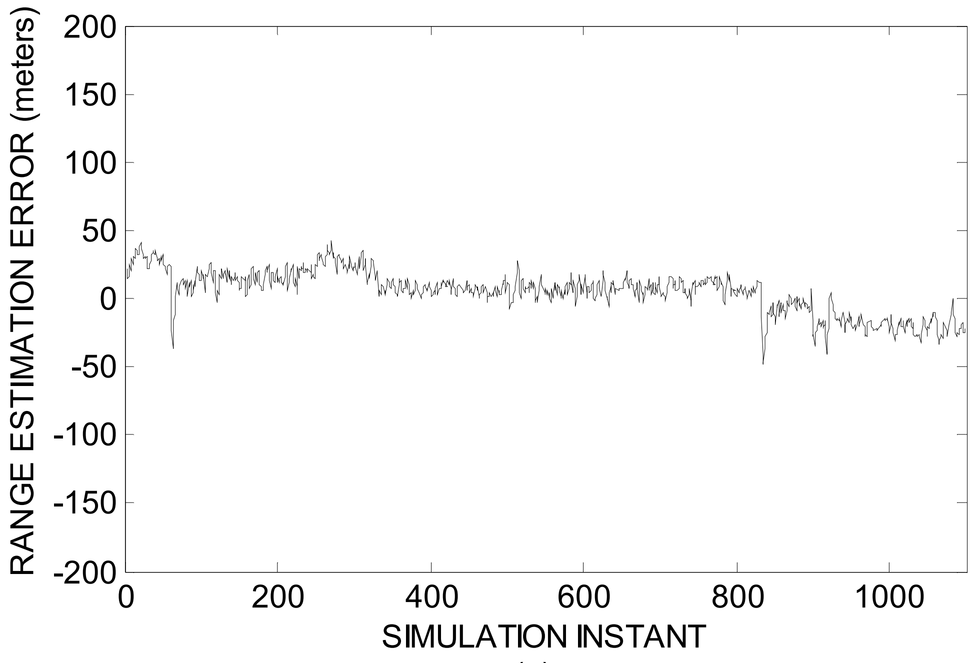
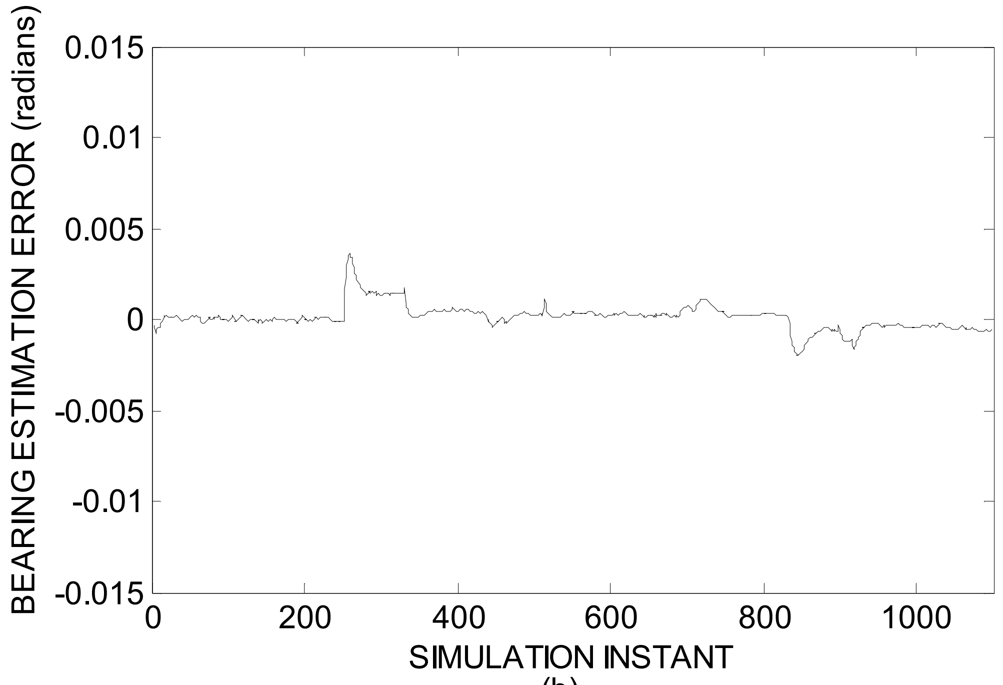


Fig. 27. Percentage battery charge remaining at the end of 1100 simulation instants in the sensor nodes (a), cluster nodes (b), and the four master nodes (c) for scenario 3. The circled region in (a) shows the effect of scheme CC4 used for localized optimization routing. When the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster nodes rotated the routing functions among different sensor nodes, resulting in uniform battery usage at the sensor nodes.



(a)



(b)

Fig. 28. Range estimation error in meters (a) and bearing estimation error in radians (b) for scenario 3.

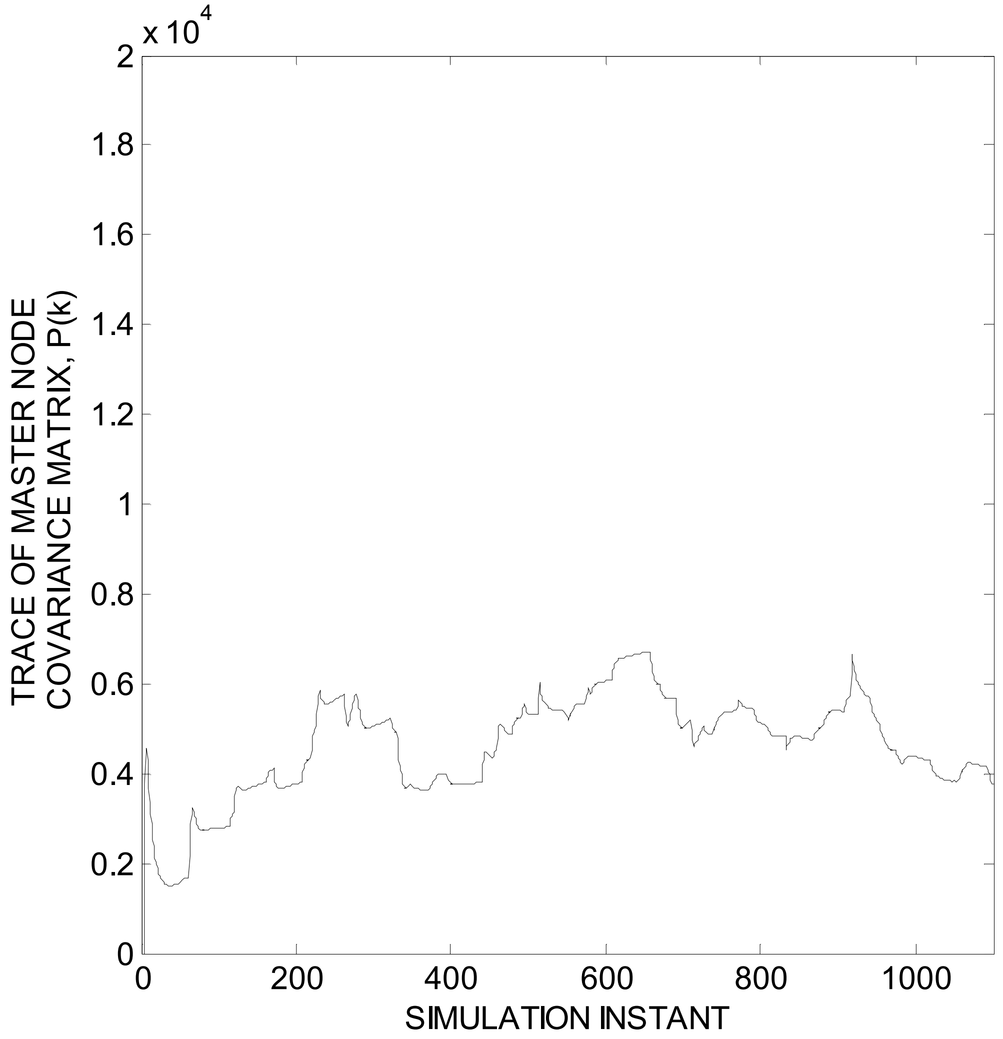


Fig. 29. Trace of the master node *P(k)* for scenario 3.

4) Scenario 4a: Figs. 30-32 show the simulation results for target tracking in scenario 4a. In scenario 4a, there were four master nodes present in the system, but only one was active at any time. Similar to scenario 3, scheme CC1 was used for rotating the master node functions among redundant nodes, and scheme CC4 was used for localized optimization routing in scenario 4a. In CC4, when the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes (circled region in Fig. 30a). In addition to schemes CC1 and CC4, scheme CCT with the *Th* set to 1410 was also implemented in scenario 4a. Since the trace of the master node *P(k+n)* was always above *Th* (Fig. 32b), the master node performed measurement updates at every simulation instant in order to meet the desired tracking performance. Hence, the results of battery consumption (Fig. 30), tracking errors (Fig. 31), and trace of the master node *P(k)* (Fig. 32) for scenario 4a were similar to the results obtained for scenario 3.

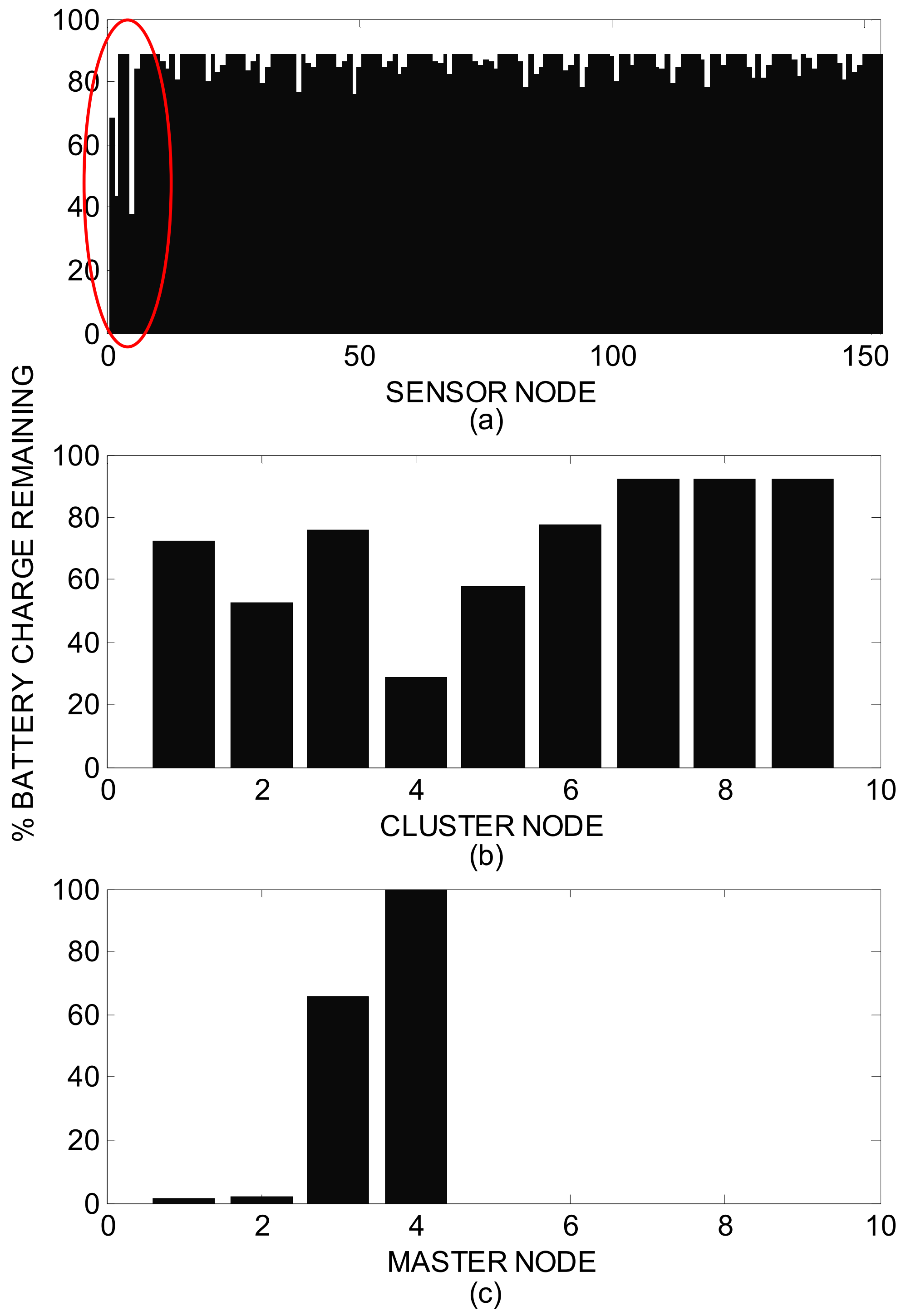
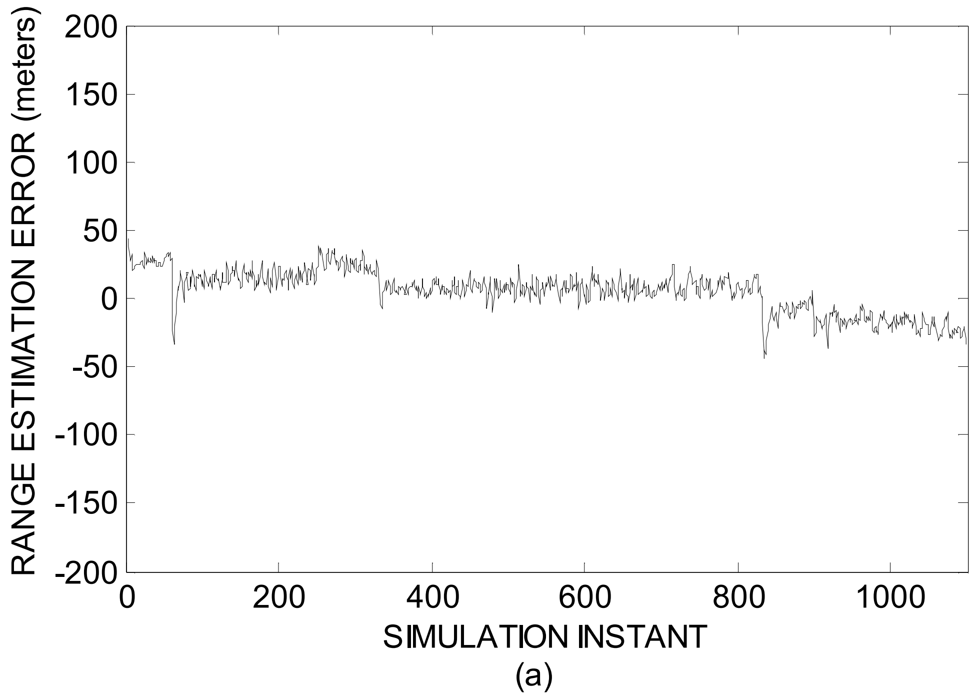


Fig. 30. Percentage battery charge remaining at the end of 1100 simulation instants in the sensor nodes (a), cluster nodes (b), and the four master nodes (c) for scenario 4a. The circled region in (a) shows the effect of scheme CC4 used for localized optimization routing. When the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes.



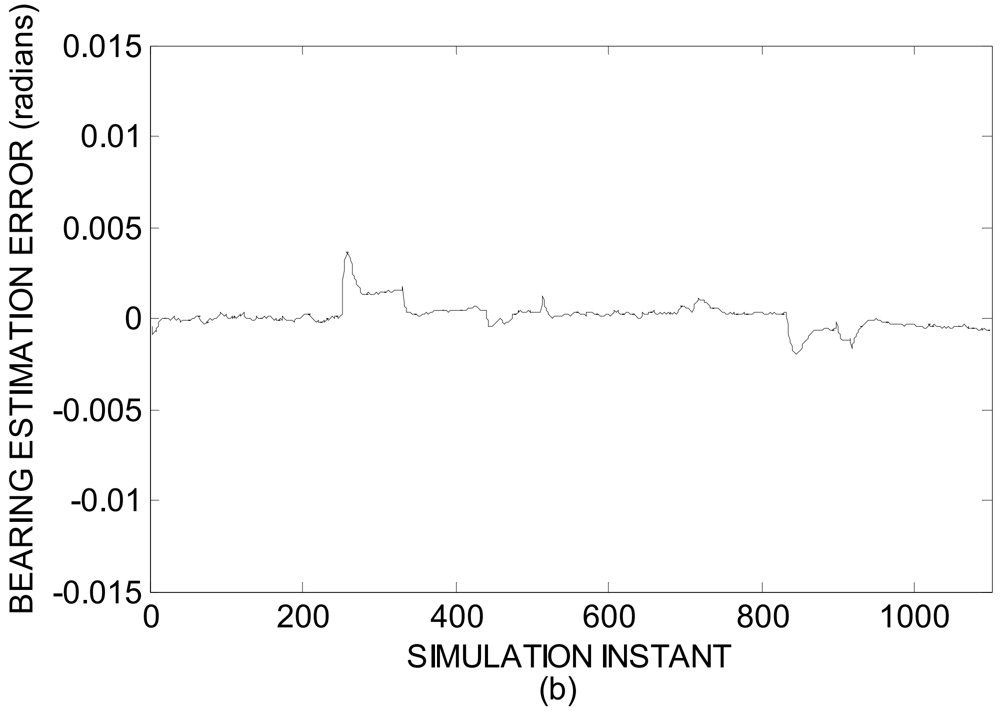
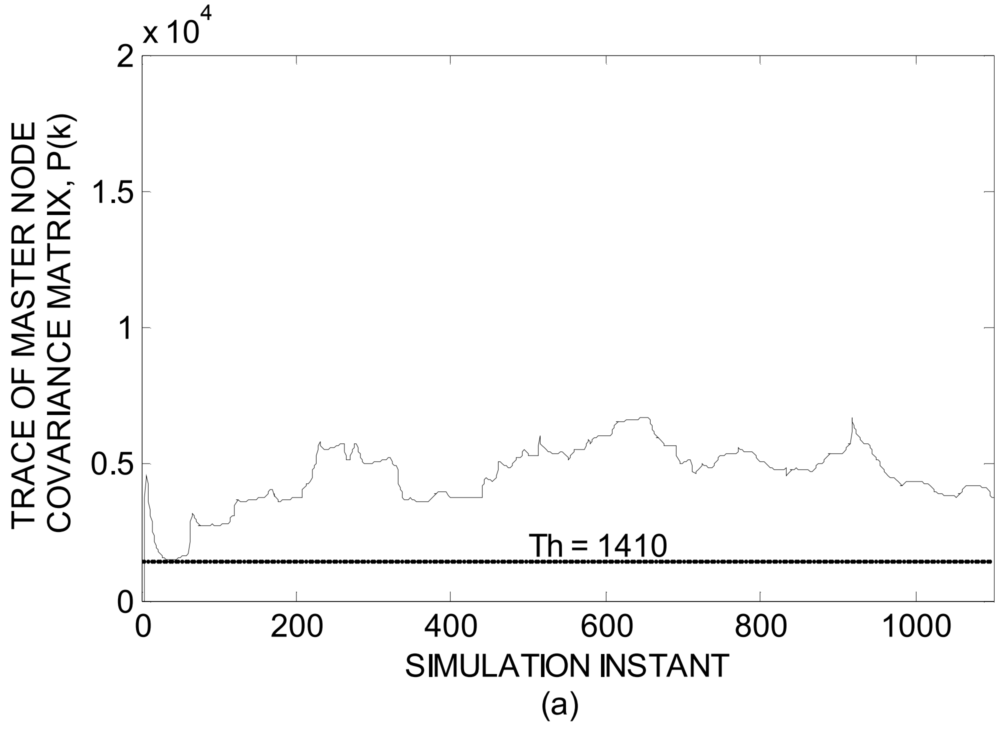


Fig. 31. Range estimation error in meters (a) and bearing estimation error in radians (b) for scenario 4a.



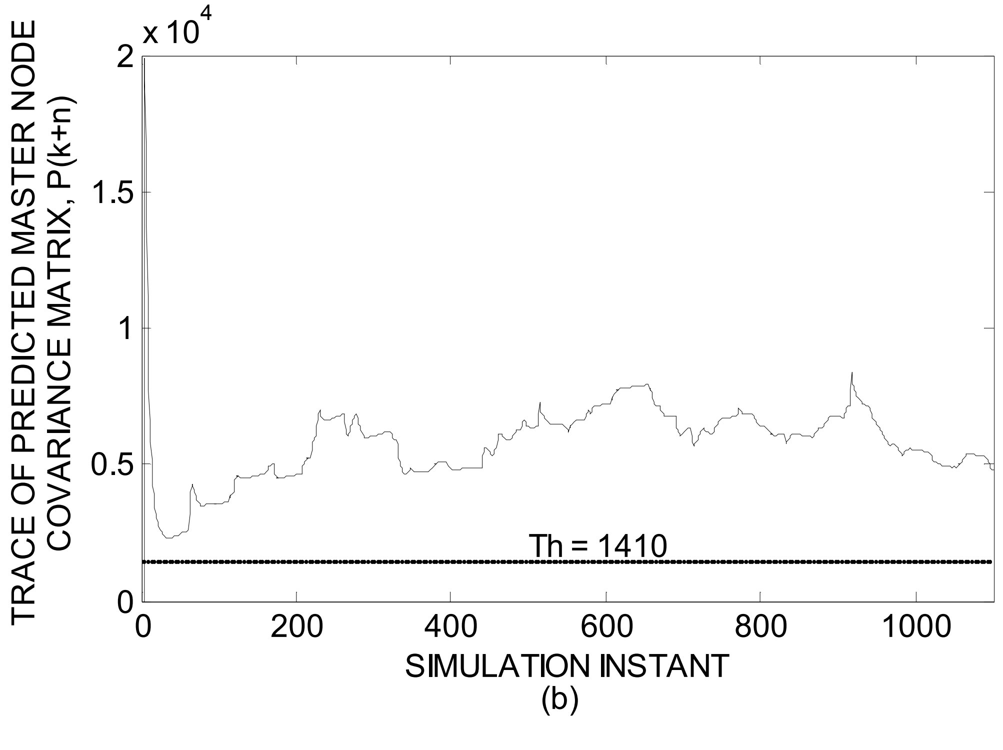


Fig. 32. Trace of the master node *P(k)* (a) and trace of the predicted master node *P(k+n)* (b) for scenario 4a. The matrix *P(k+n)* is the matrix *P(k)* predicted two time steps ahead *(n=2)*.

5) Scenario 4b: Figs. 33-35 show the simulation results for target tracking in scenario 4b. Similar to scenario 4a, schemes CC1, CC4, and CCT were implemented in scenario 4b. In CC4, when the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster nodes rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes (circled region in Fig. 33a). In scenario 4b, CCT was implemented with the *Th* set to 4230. Initially, until around 150 simulation instants, the trace of the master node *P(k+n)* was below *Th* for some simulation instants (Fig. 34b). Thus, CCT was able to reduce the rate at which local estimates were communicated to the master node and measurement updates were performed at the master node, while still achieving the desired tracking performance. Beyond 150 simulation instants, the trace of the master node *P(k+n)* was always above *Th*, even though a measurement update occurred at every simulation instant; therefore, beyond 150 simulation instants, CCT with the *Th* set to 4230 was unable to conserve resources in the network.

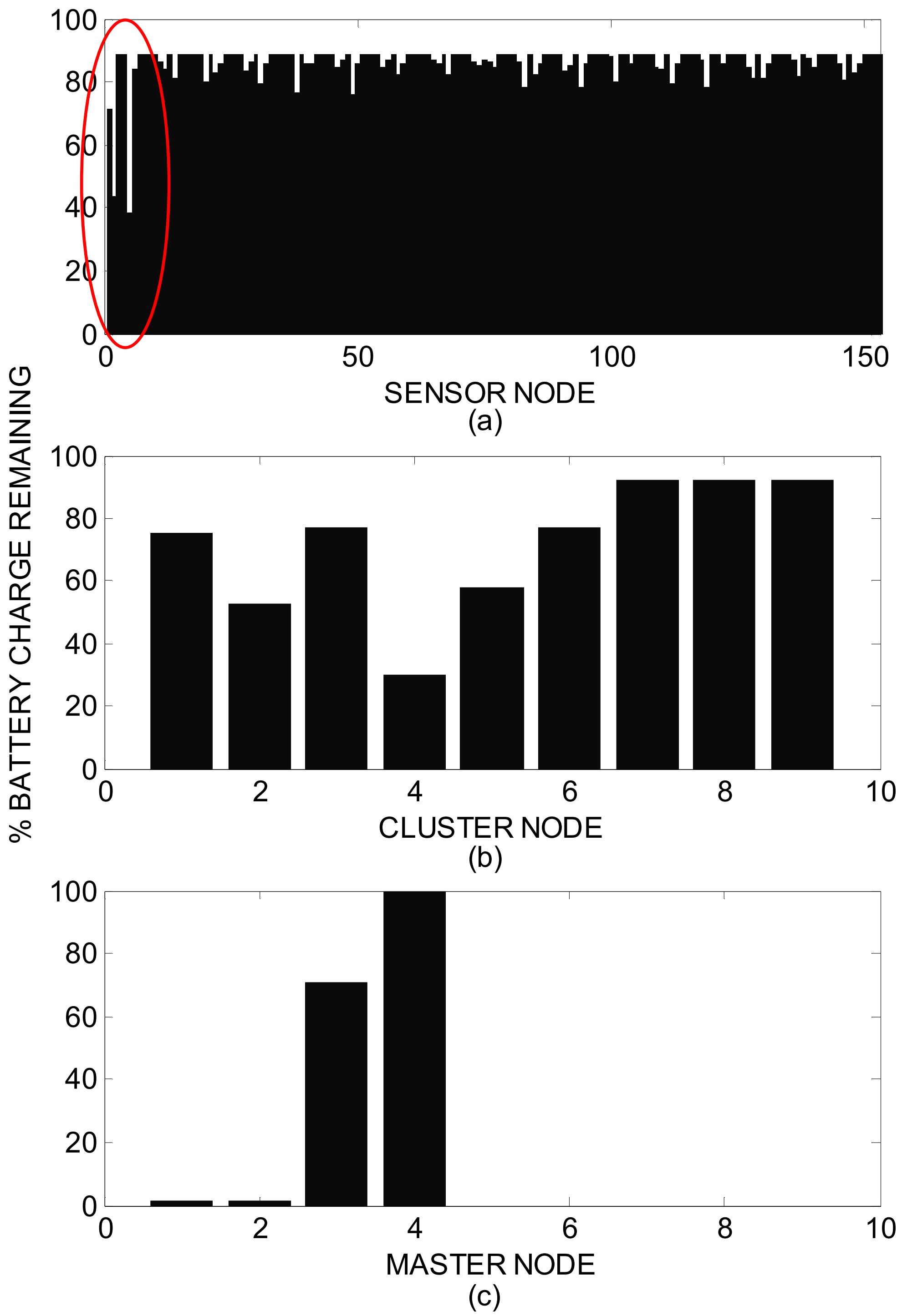
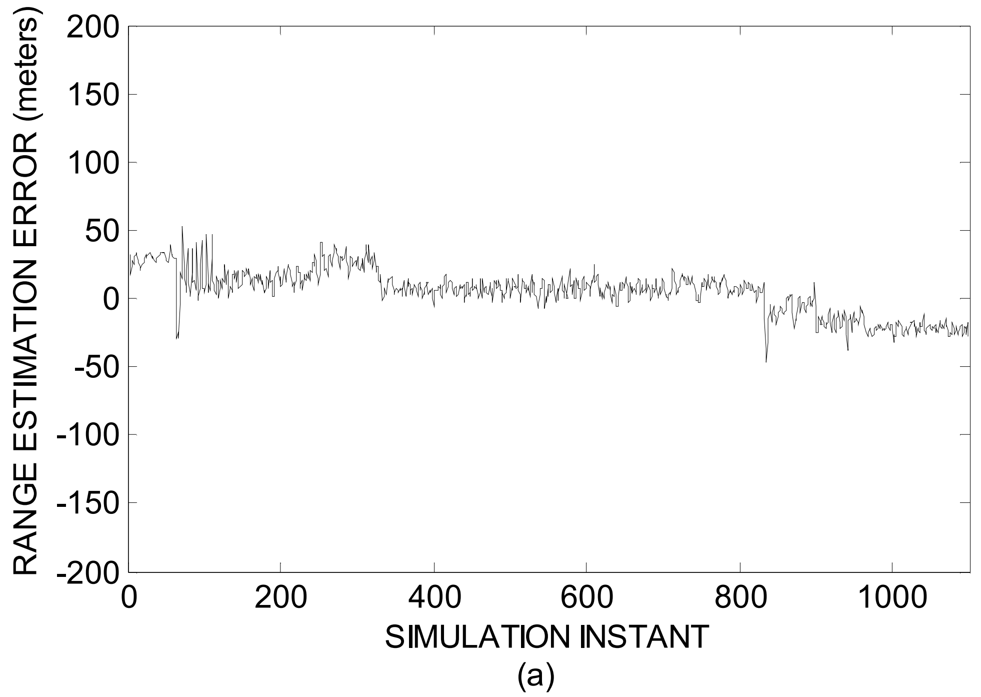


Fig. 33. Percentage battery charge remaining at the end of 1100 simulation instants in the sensor nodes (a), cluster nodes (b), and the four master nodes (c) for scenario 4b. The circled region in (a) shows the effect of scheme CC4 used for localized optimization routing. When the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster nodes rotated the routing function among different sensor nodes, resulting in more uniform battery usage at the sensor nodes.



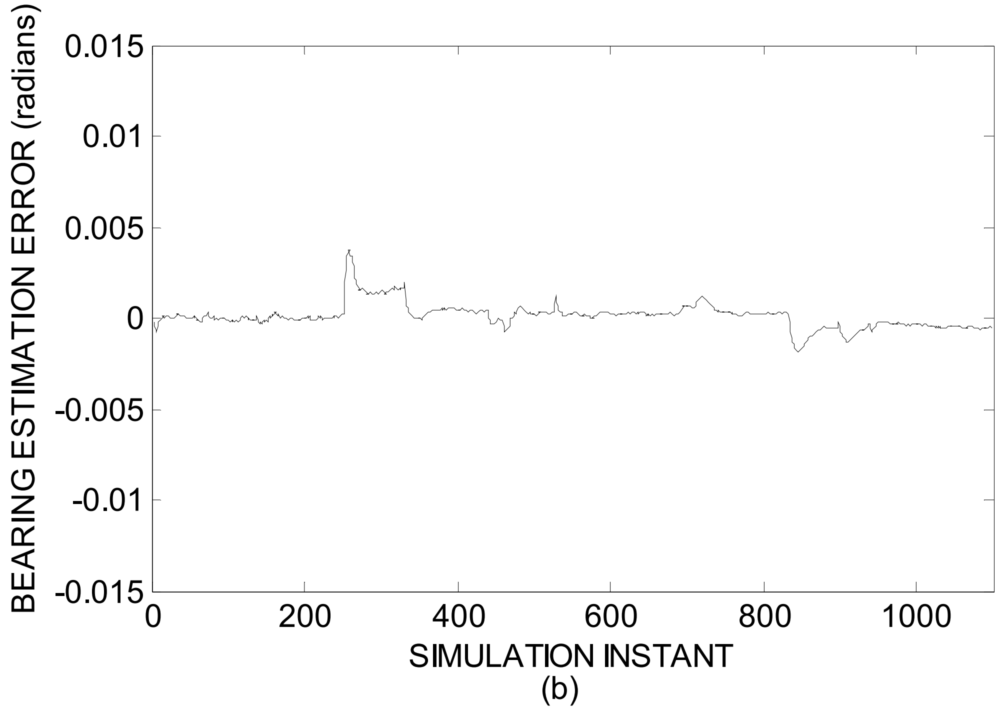
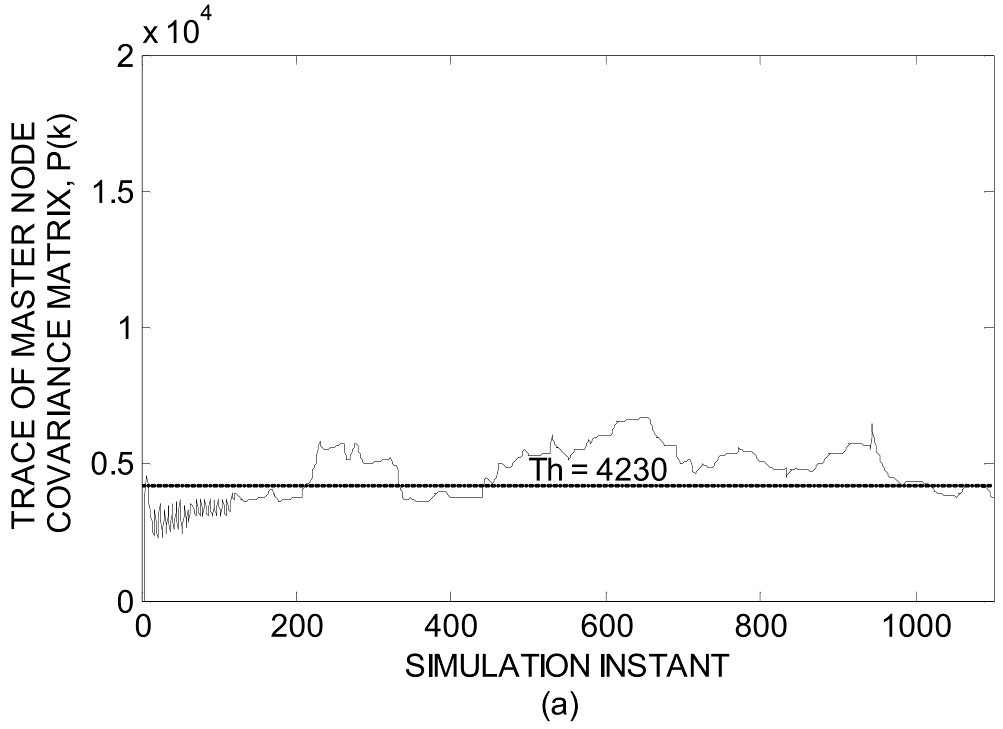


Fig. 34. Range estimation error in meters (a) and bearing estimation error in radians (b) for scenario 4b.



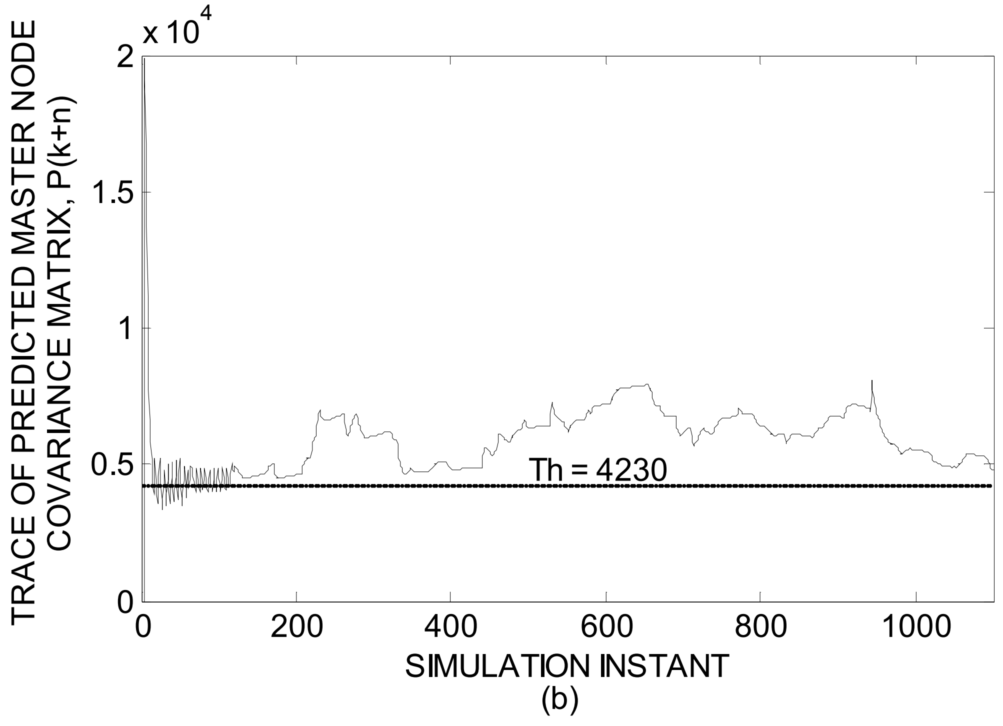


Fig. 35. Trace of the master node *P(k)* (a) and trace of the predicted master node *P(k+n)* (b) for scenario 4b. The matrix *P(k+n)* is the matrix *P(k)* predicted two time steps ahead *(n=2)*.

6) Scenario 4c: Figs. 36-38 show the simulation results for target tracking in scenario 4c. Similar to scenario 4a, schemes CC1, CC4, and CCT were implemented in scenario 4c. In CC4, when the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster node rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes (circled region in Fig. 36a). In scenario 4c, CCT was implemented with the *Th* set to 7050. Whenever the trace of the master node *P(k+n)* was below *Th* (Fig. 38b), CCT was able to reduce the rate at which local estimates were communicated from the cluster nodes to the master node and measurement updates were performed at the master node, while still achieving the desired tracking performance. Since local estimates were communicated less frequently from the cluster nodes to the master nodes, these nodes were able to conserve their battery charge (Fig. 36). Fewer measurement updates resulted in large range and bearing estimation errors due to the large *Th* in CCT (Fig. 37).

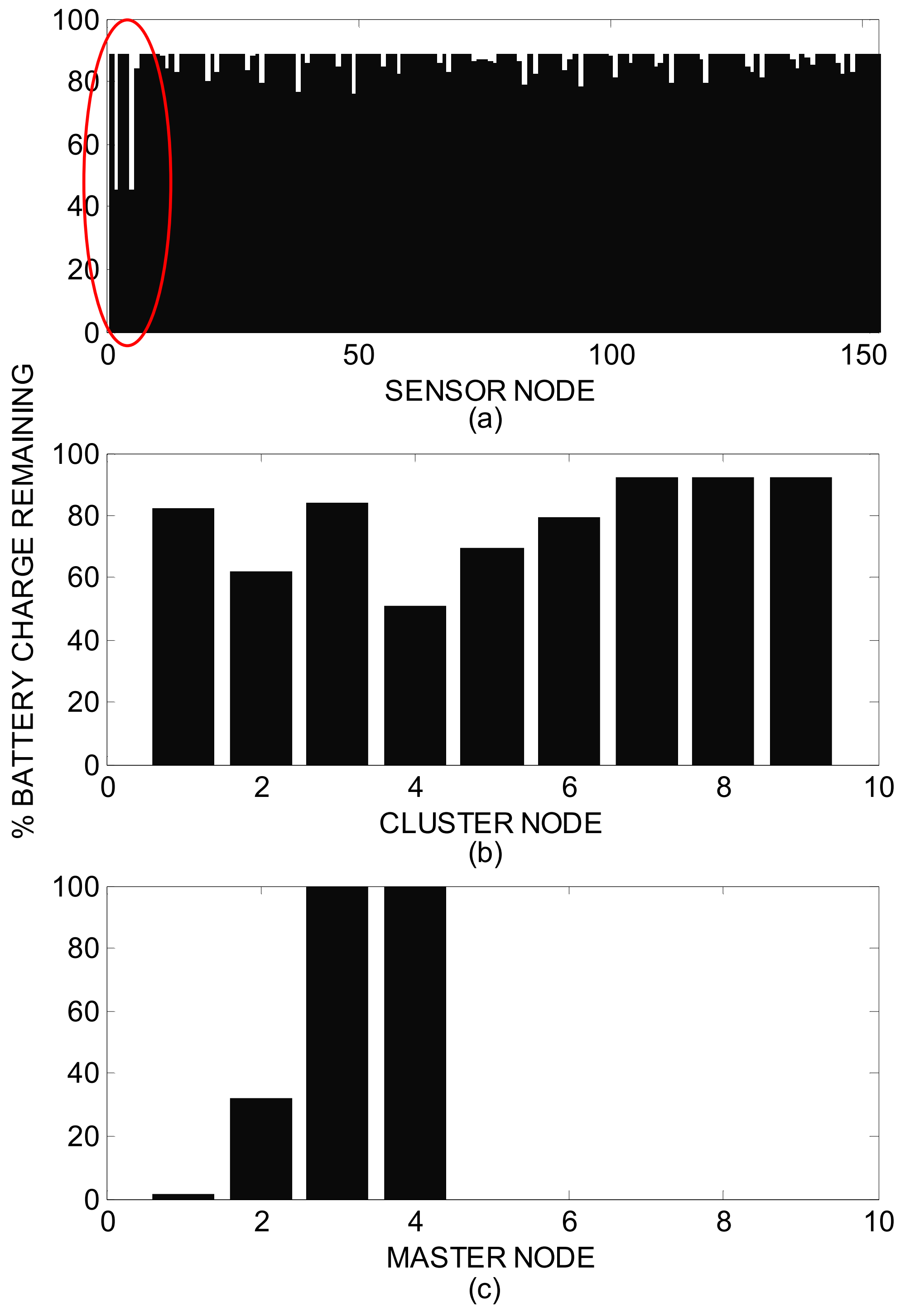
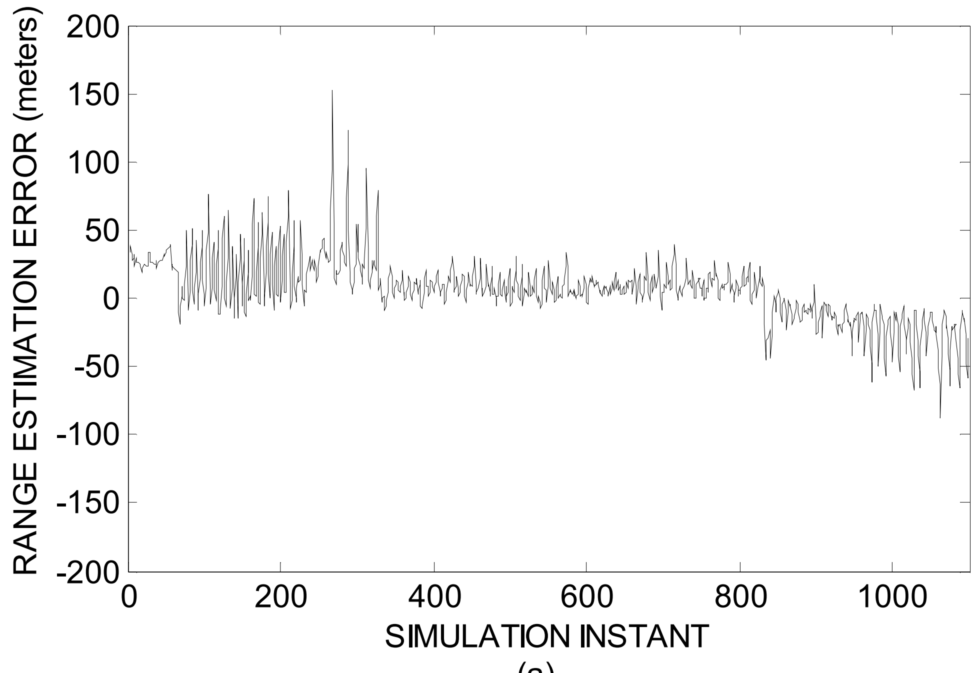


Fig. 36. Percentage battery charge remaining at the end of 1100 simulation instants in the sensor nodes (a), cluster nodes (b), and the four master nodes (c) for scenario 4c. The circled region in (a) shows the effect of scheme CC4 used for localized optimization routing. When the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster nodes rotated the routing function among different sensor nodes, resulting in more uniform battery usage at the sensor nodes.



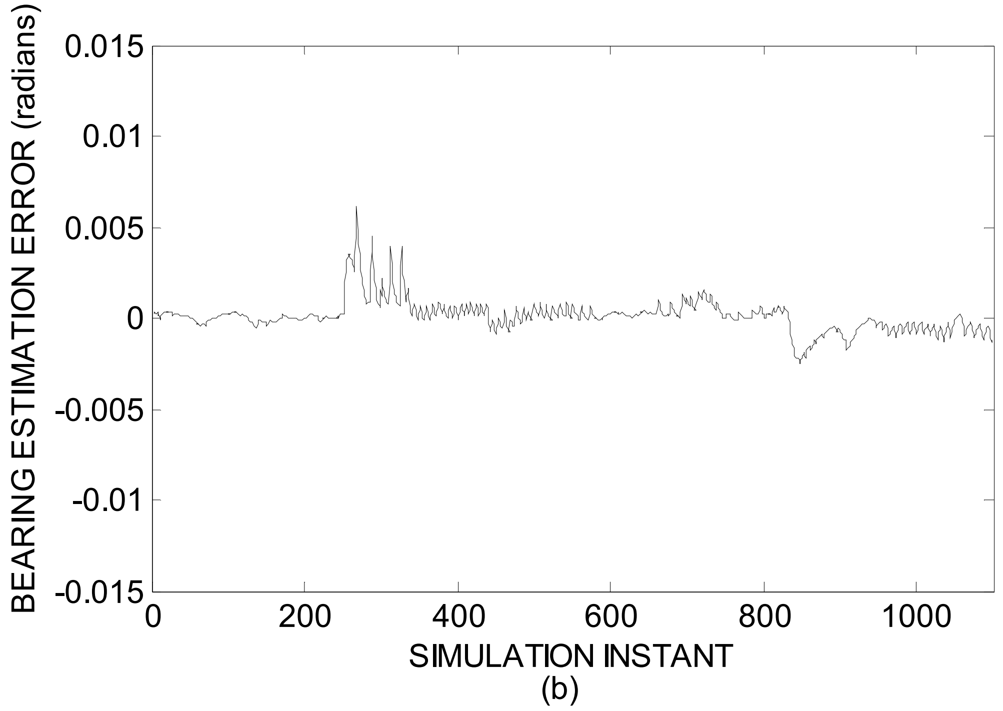
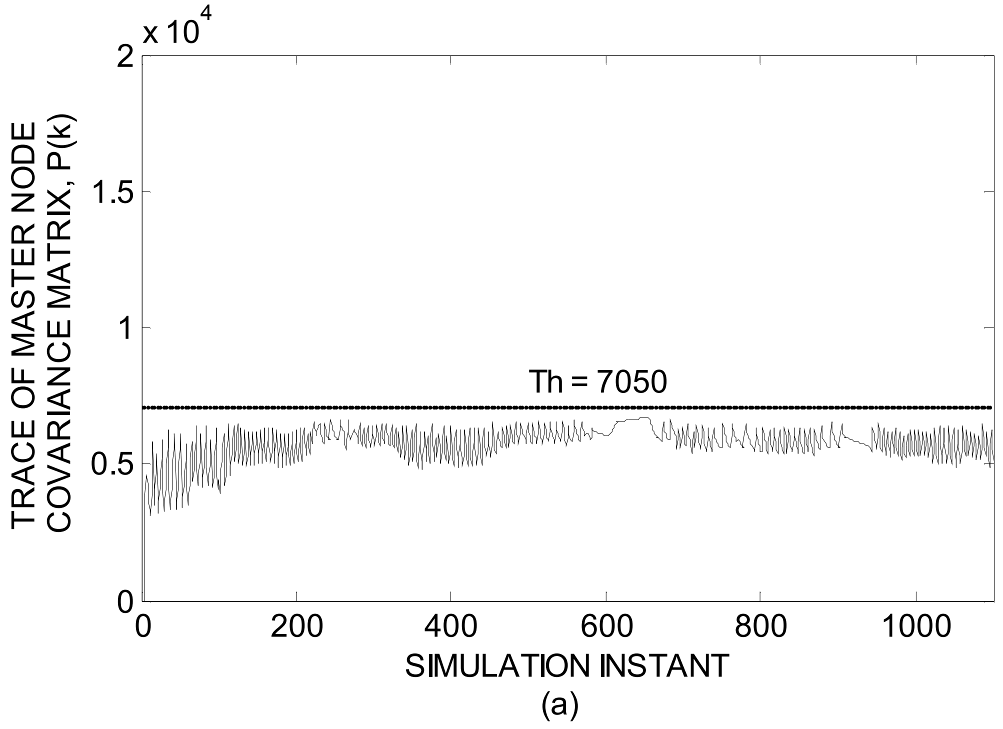


Fig. 37. Range estimation error in meters (a) and bearing estimation error in radians (b) for scenario 4c.



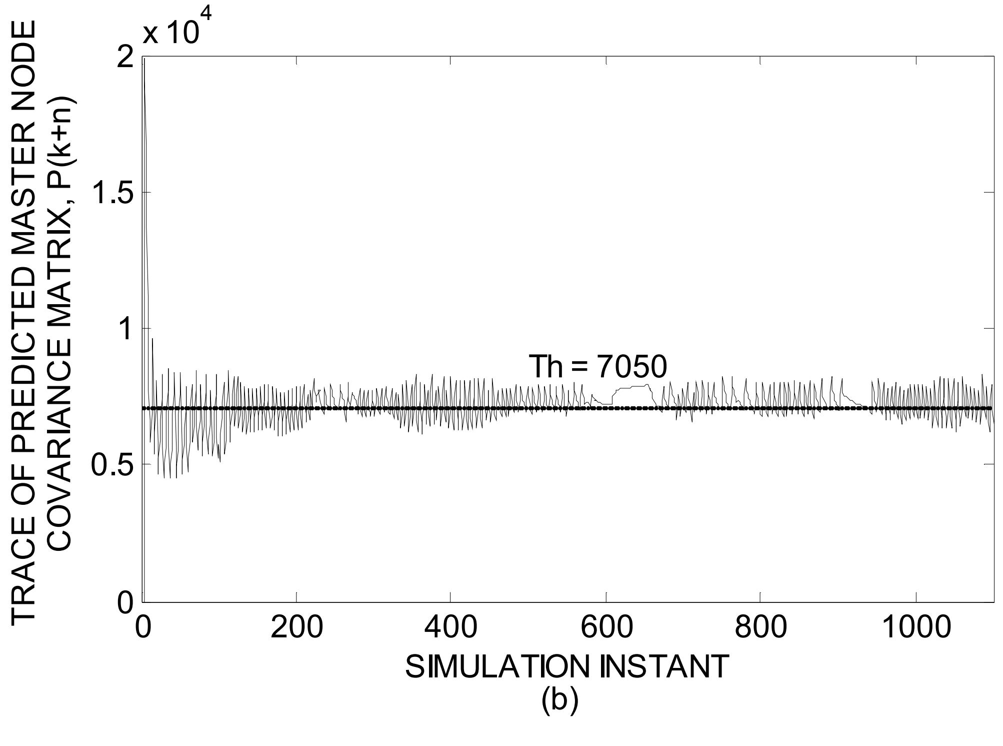


Fig. 38. Trace of the master node *P(k)* (a) and trace of the predicted master node *P(k+n)* (b) for scenario 4c. The matrix *P(k+n)* is the matrix *P(k)* predicted two time steps ahead *(n=2)*.

7) Scenario 4d: Figs. 39-41 show the simulation results for target tracking in scenario 4d. Similar to scenario 4a, schemes CC1, CC4, and CCT were implemented in scenario 4d. In CC4, when the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster node rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes; however, since none of the sensor nodes had their battery charge fall below 50%, scheme CC4 was never used in scenario 4d (circled region in Fig. 39a). In scenario 4d, CCT was implemented with the *Th* set to 14100. Whenever the trace of the master node *P(k+n)* was below *Th* (Fig. 40b), CCT was able to reduce the rate at which local estimates were communicated from the cluster nodes to the master node and measurement updates were performed at the master node, while still achieving the desired tracking performance. Since local estimates were communicated less frequently from the cluster nodes to the master nodes, these nodes were able to conserve their battery charge (Fig. 39). Fewer measurement updates resulted in large range and bearing estimation errors due to the large *Th* in CCT (Fig. 40). Due to fewer measurement updates, the master node battery charge was conserved, and only one master node was partially used during the simulation (Fig. 39c).

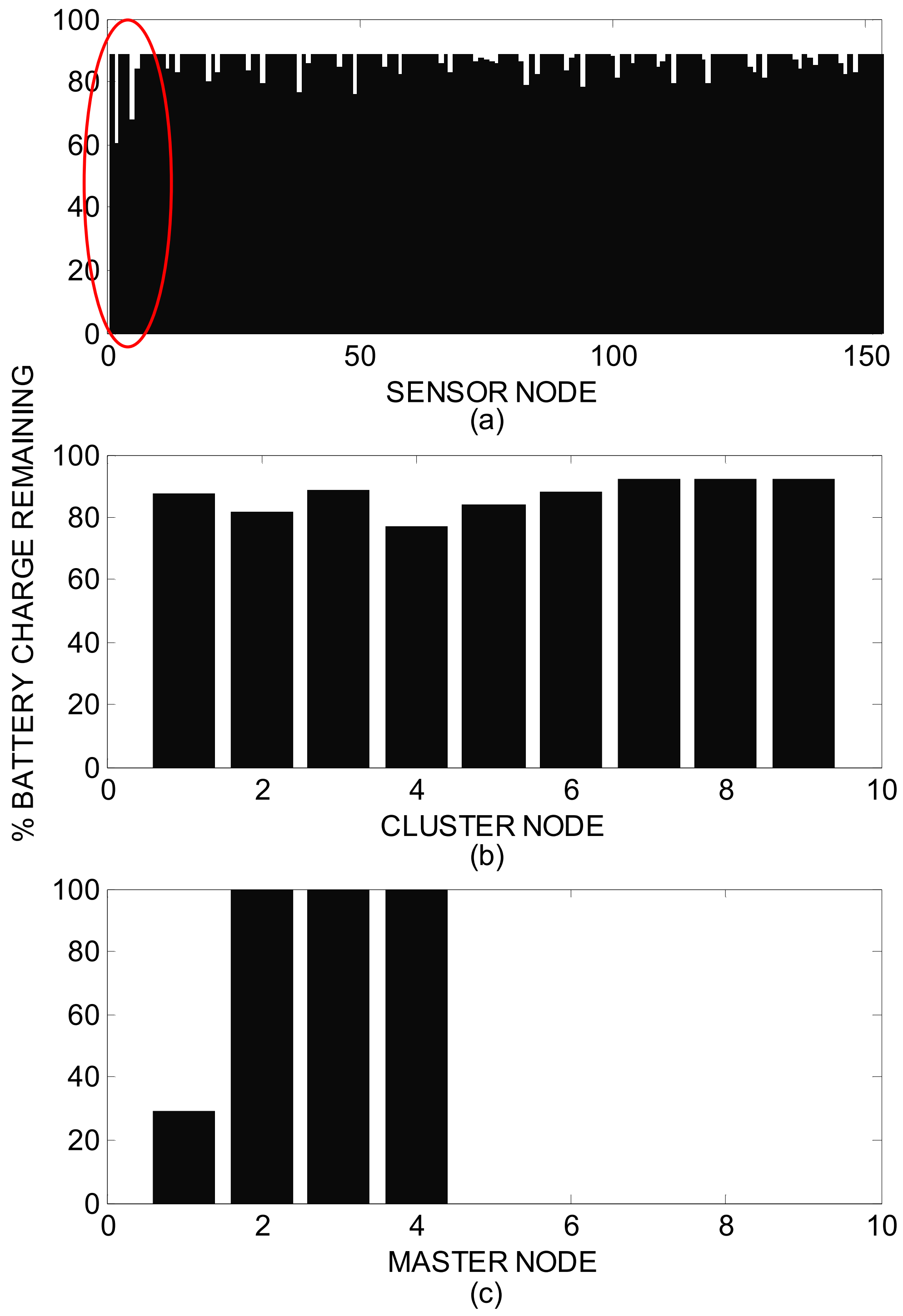
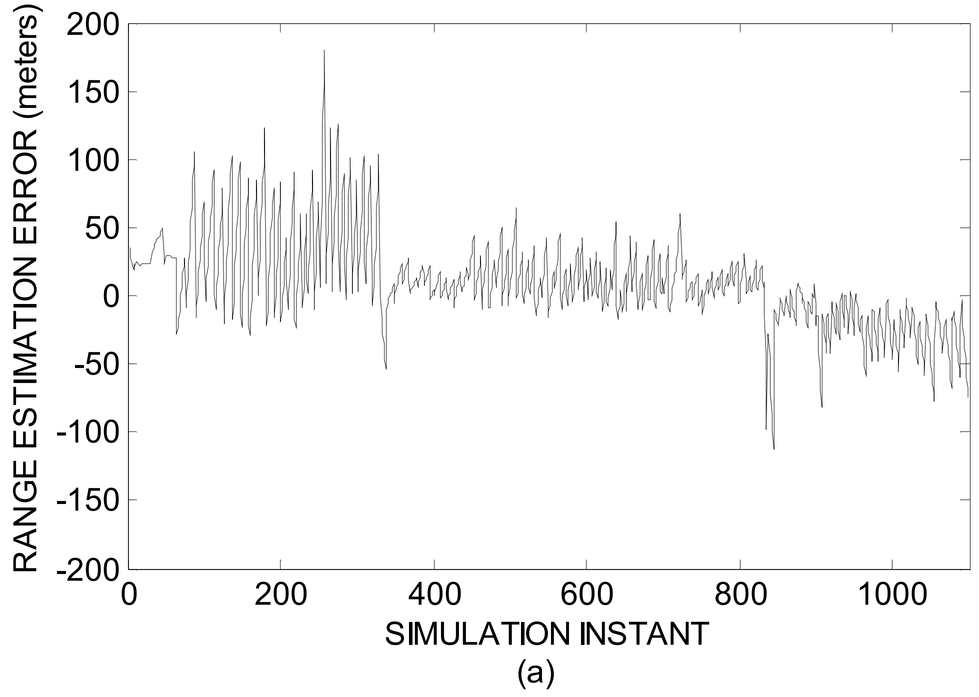


Fig. 39. Percentage battery charge remaining at the end of 1100 simulation instants in the sensor nodes (a), cluster nodes (b), and the four master nodes (c) for scenario 4d. In scenario 4d, no sensor had battery charge below 50% and hence scheme CC4 for localized optimization routing was never used.



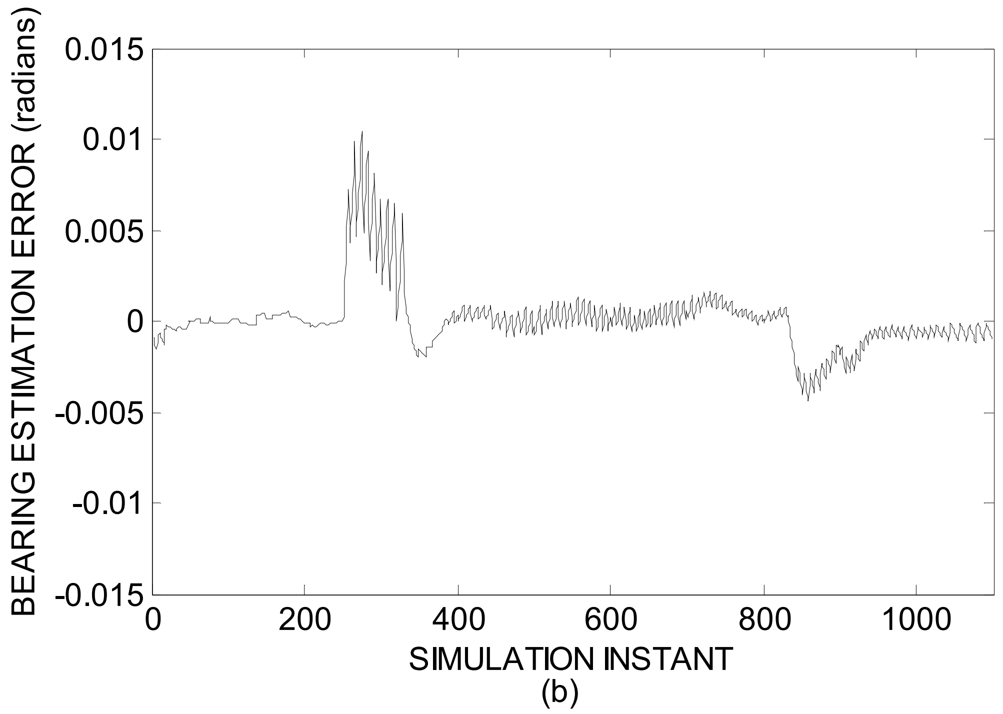
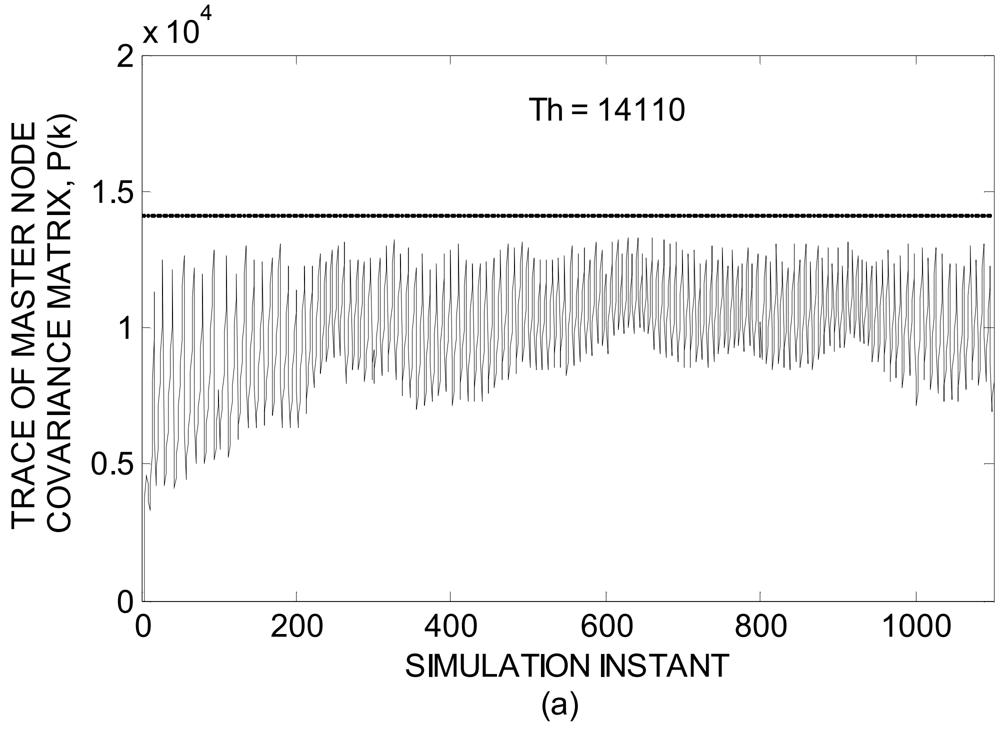


Fig. 40. Range estimation error in meters (a) and bearing estimation error in radians (b) for scenario 4d.



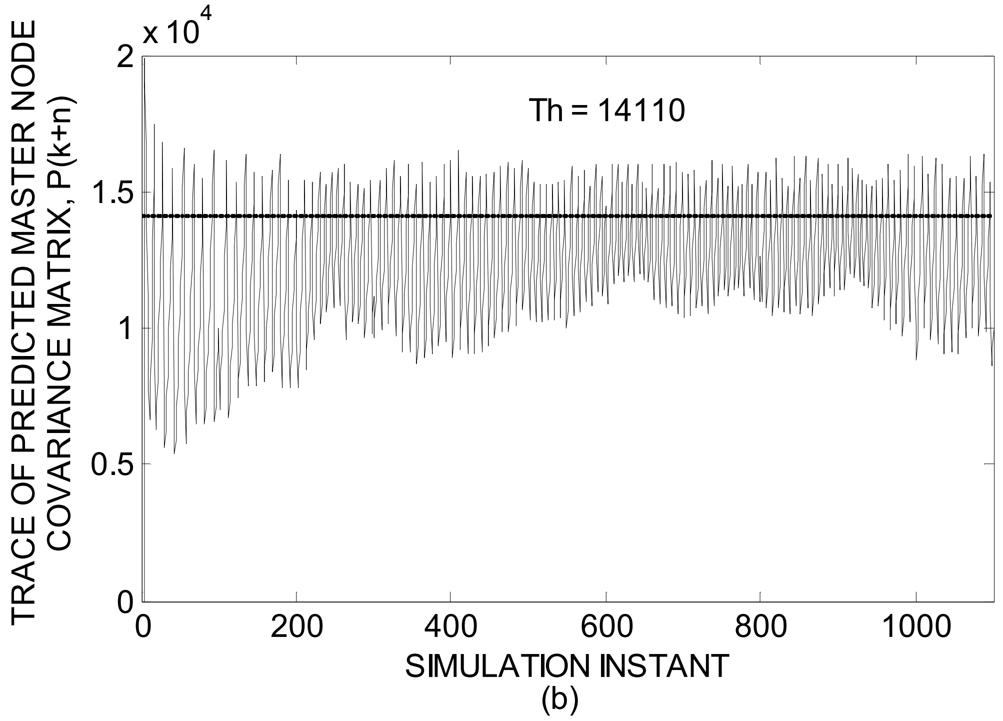


Fig. 41. Trace of the master node *P(k)* (a) and trace of the predicted master node *P(k+n)* (b) for scenario 4d. The matrix *P(k+n)* is the *P(k)* matrix predicted two time steps ahead *(n=2)*.

8) Scenario 5: Fig. 42 shows the simulation results for target tracking in scenario 5. Similar to scenarios 4a-4d, schemes CC1, CC4, and CCT were implemented in scenario 5. In CC4, when the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster nodes rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes (circled region in Fig. 42a). In scenario 5, CCT was implemented with the *Th* set to 7050 similar to scenario 4c. Hence, tracking results were similar to those obtained for scenario 4c (Figs. 37 and 38). In addition to CC1, CC4, and CCT, schemes CC2 and CC3 were also implemented in scenario 5. Sensor and cluster nodes that were far away from the target were switched off in schemes CC2 and CC3 in order to conserve battery charge (Fig. 42). Sensor and cluster nodes that were switched off during the entire simulation duration had almost 100% of their battery remaining at the end of the simulation (Fig. 42).

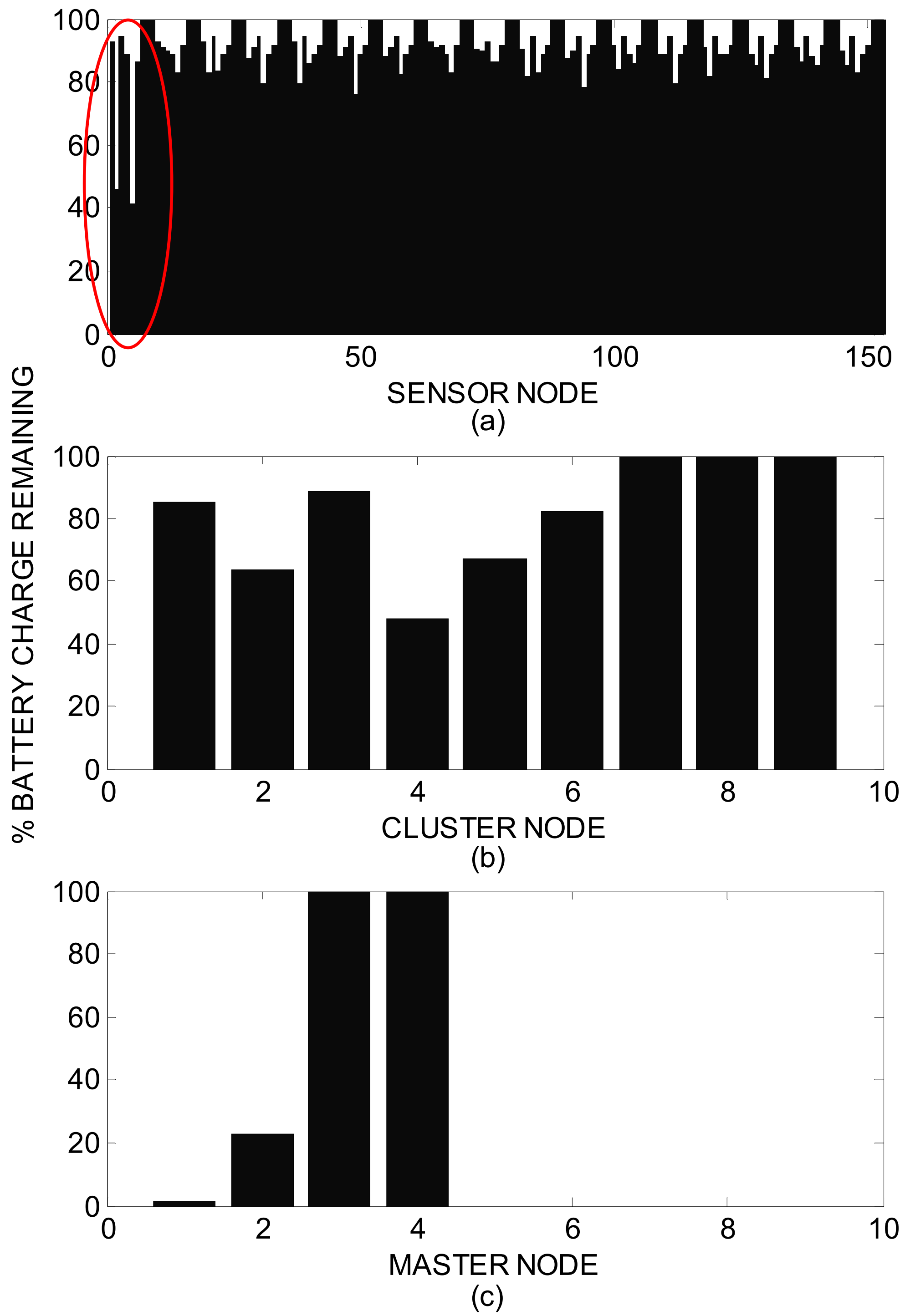


Fig. 42. Percentage battery charge remaining at the end of 1100 simulation instants in the sensor nodes (a), cluster nodes (b), and the four master nodes (c) for scenario 5. The circled region in (a) shows the effect of scheme CC4 used for localized optimization routing. When the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster nodes rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes.

9) Scenario 6: Figs. 43-45 show the simulation results for target tracking in scenario 6. In scenario 6, the master and cluster agent functions were stacked on the same physical hardware. Because of this stacking, there was an increase in the number of nodes in the system that could perform the master node functions without the need for extra master node hardware. There was an increase in the battery charge drainage of the master/cluster node when the node performed both the master and cluster functions. Rotating the master node functions among the nine master/cluster nodes reduced the burden on any one master/cluster node and balanced the resource utilization in the cluster nodes (Fig 43b). Sensors and clusters that were far away from the target were switched off in schemes CC2 and CC3 in order to conserve battery charge. Sensor nodes that were switched off during the entire simulation duration had almost 100% of their battery remaining at the end of the simulation (Fig. 43a). Scheme CC4 was used for localized optimization routing. In CC4, when the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster agent rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes (circled region in Fig. 43a). In scenario 6, CCT is implemented with the *Th* set to 7050. Whenever the trace of the master node *P(k+n)* was below *Th* (Fig. 45b), CCT was able to reduce the rate at which local estimates were communicated from the cluster nodes to the master node and measurement updates were performed at the master node, while still achieving the desired tracking performance.

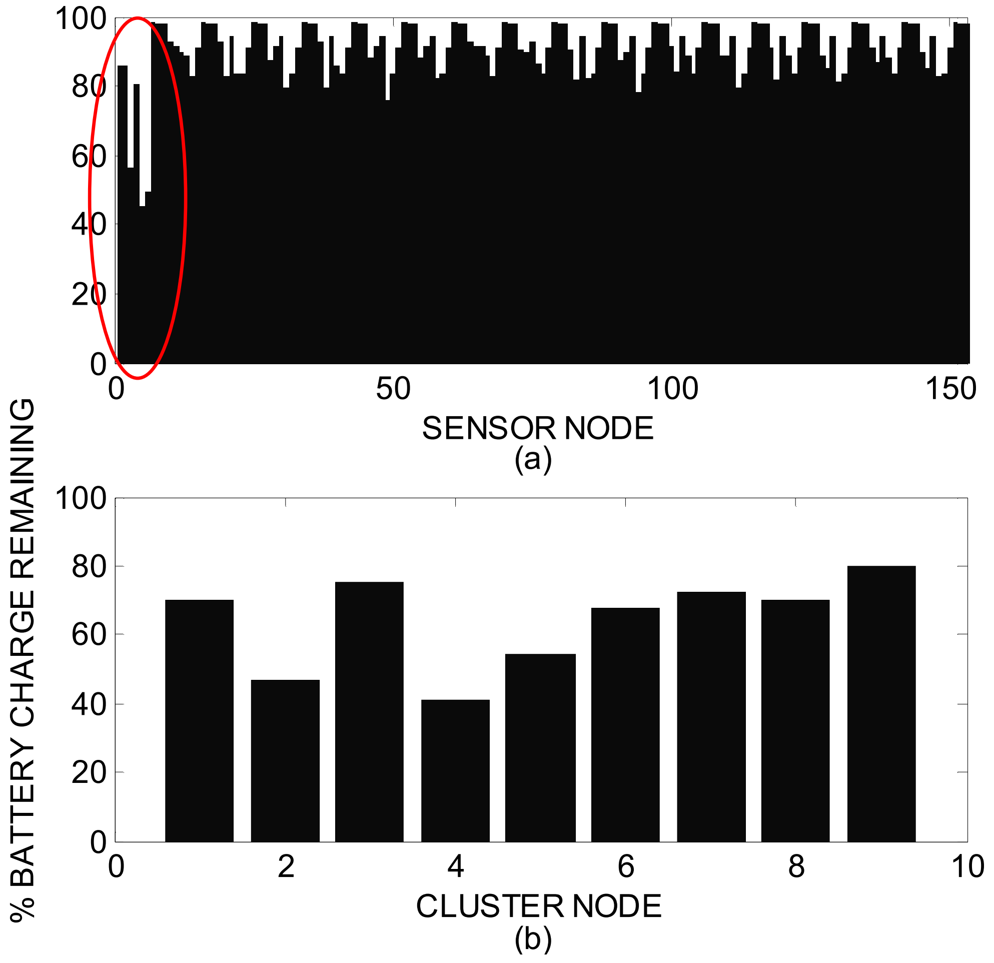
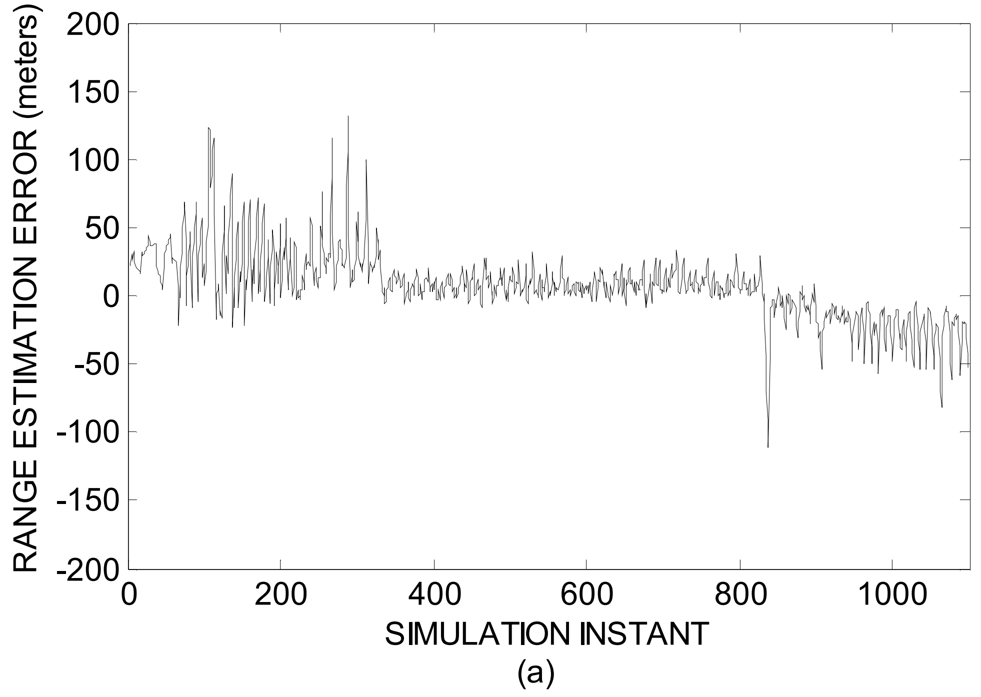


Fig. 43. Percentage battery charge remaining at the end of 1100 simulation instants in the sensor nodes (a) and cluster nodes (b) for scenario 6. Stacking functional agents enables the cluster nodes to accomplish the tasks of the master nodes and, hence, only the cluster is displayed for scenario 6. The circled region in (a) shows the effect of scheme CC4 used for localized optimization routing. When the battery charge of the initial set of routing sensor nodes fell below 50%, the cluster rotated the routing function among different sensor nodes, resulting in uniform battery usage at the sensor nodes.



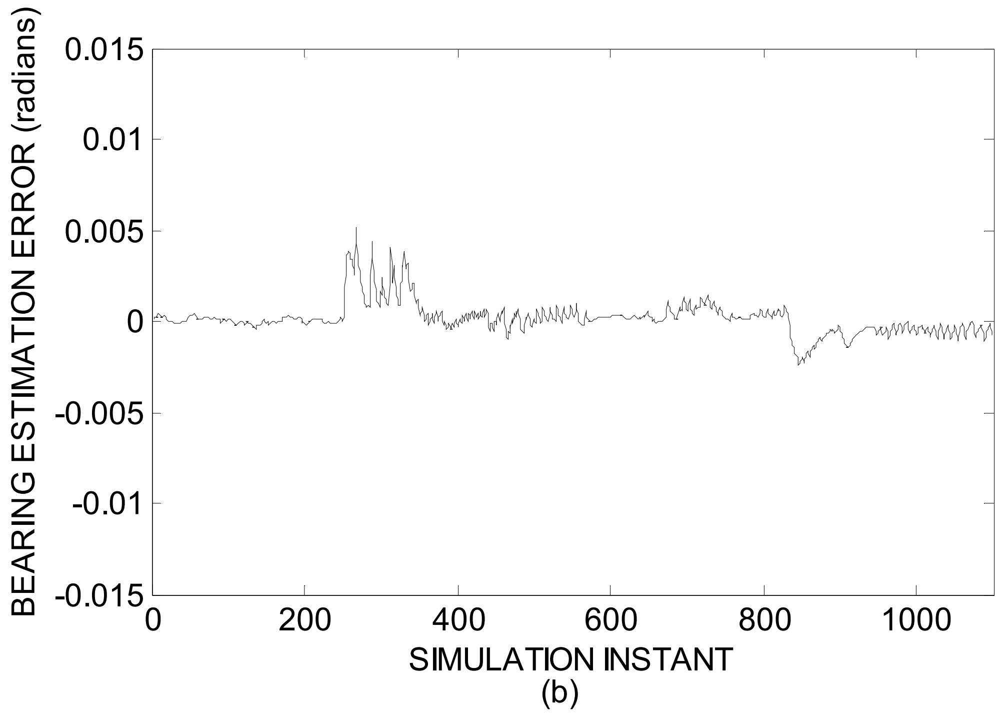
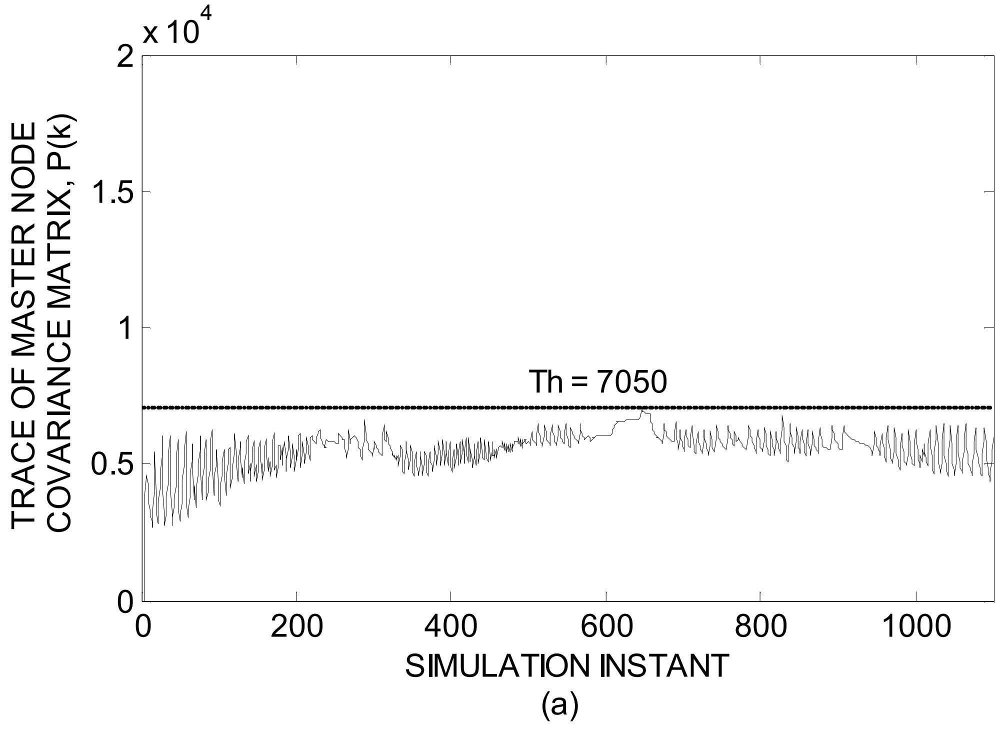


Fig. 44. Range estimation error in meters (a) and bearing estimation error in radians (b) for scenario 6.



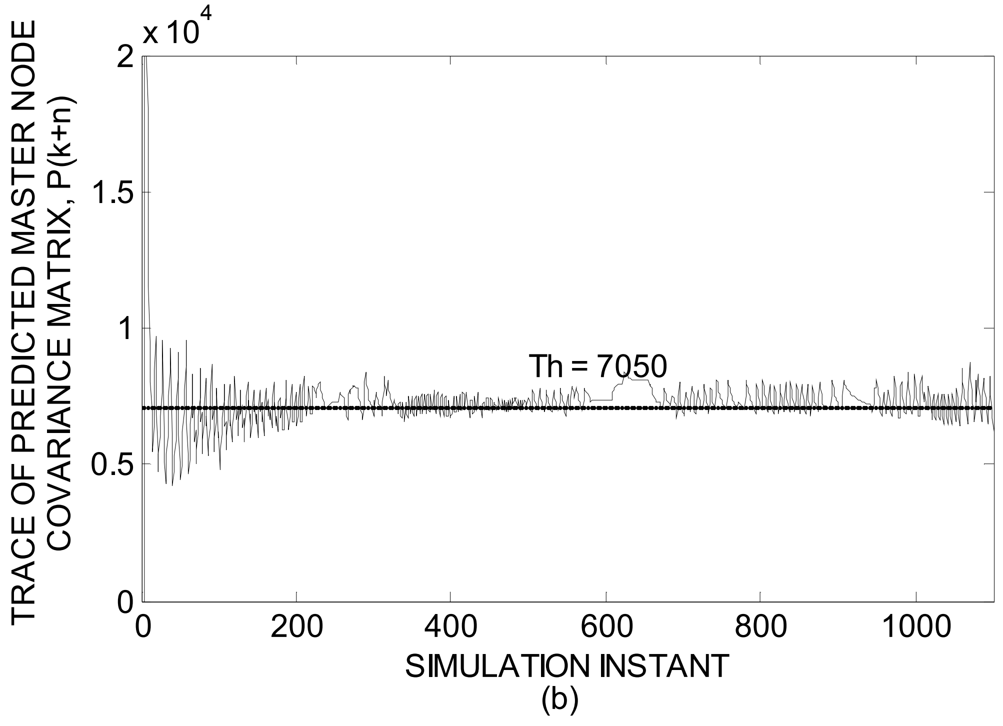


Fig. 45. Trace of the master node *P(k)* (a) and trace of the predicted master node *P(k+n)* (b) for scenario 6. The matrix *P(k+n)* is the matrix *P(k)* predicted two time steps ahead *(n=2)*.

B. Comparison Results for Single Trial Scenario Simulations

Figs. 46-50 show the comparison of performance metrics for scenarios 1-6. Fig. 46 shows the average percentage battery charge remaining, and Fig. 47 shows the minimum percentage battery charge remaining in the sensor, cluster, and master nodes at the end of simulations for each scenario (517 simulation instants for scenario 1 and 1100 simulation instants for scenarios 2-6). Fig. 48 shows the number of computations and communications per simulation instant for each scenario. Fig. 49 shows the RMS target range estimation error, and Fig. 50 shows the number of measurement updates per simulation instant at the master node for each scenario.

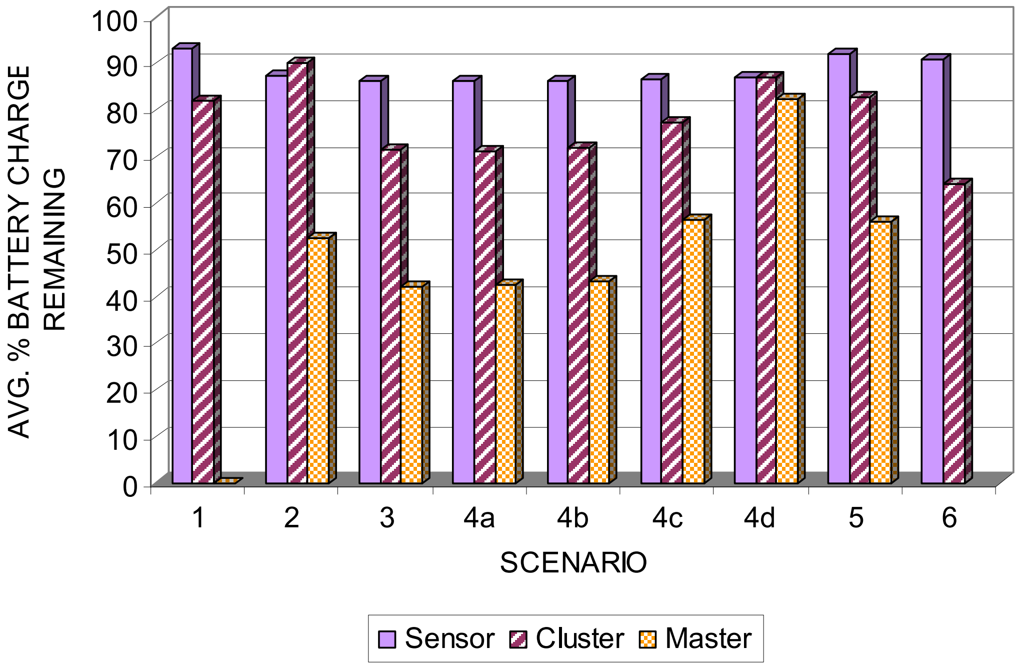


Fig. 46. Average percentage battery charge remaining in the sensor, cluster, and master nodes at the end of simulations for each agent scenario. Only one master node was present in scenarios 1 and 2. The simulation in scenario 1 stopped at simulation instant 517 due to the loss of the only master node in the network. Stacking functional agents enabled the cluster nodes to accomplish the tasks of the master nodes and, hence, there is no master displayed for scenario 6.

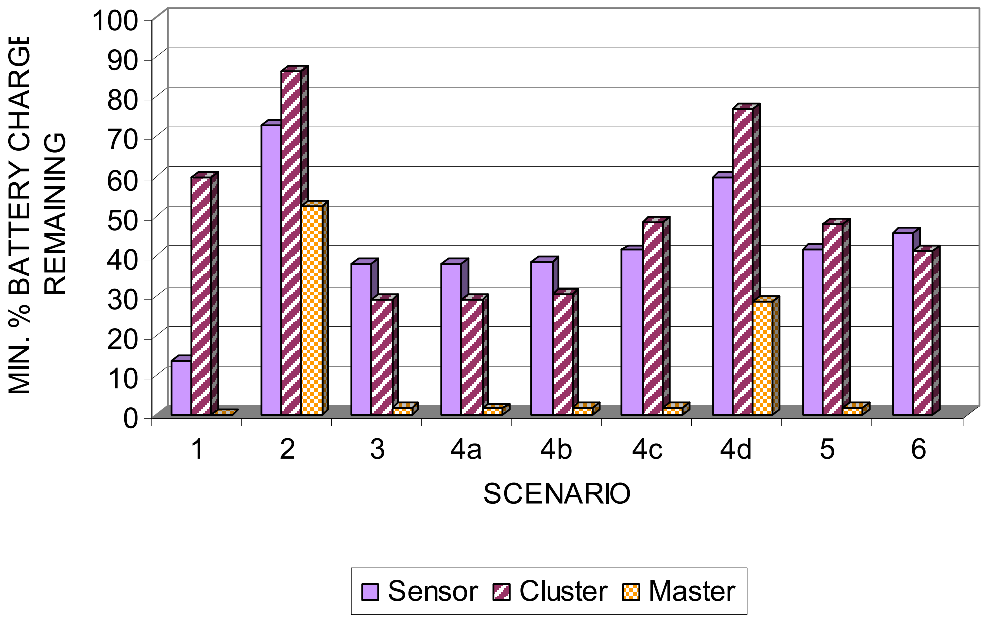


Fig. 47. Minimum percentage battery charge remaining in the sensor, cluster, and master nodes at the end of simulations for each agent scenario. Only one master node was present in scenarios 1 and 2. The simulation in scenario 1 stopped at simulation instant 517 due to the loss of the only master node in the network.

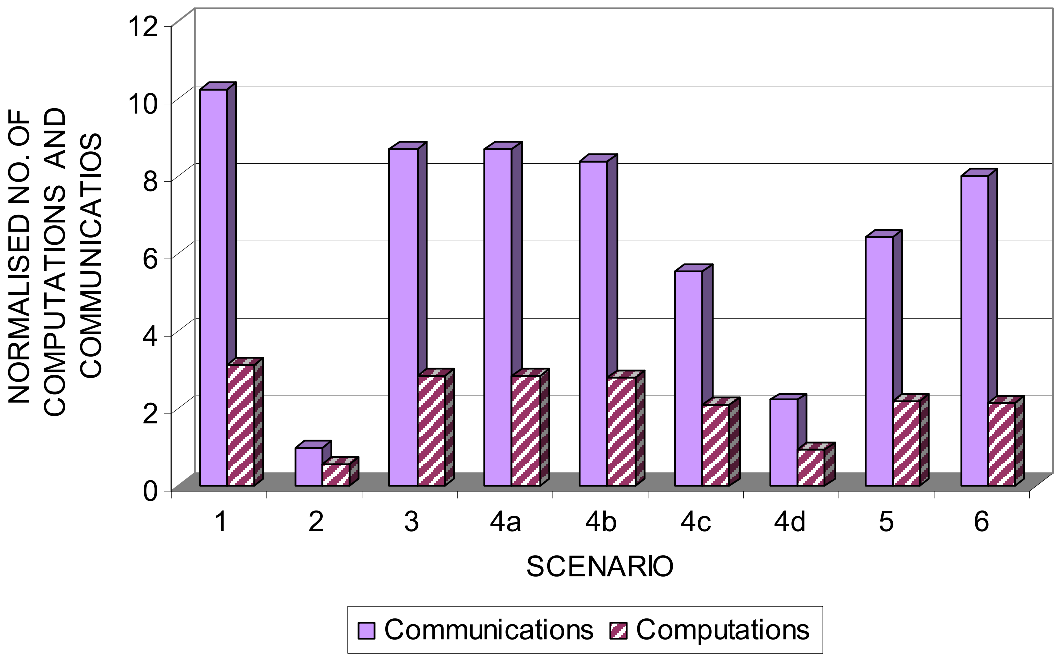


Fig. 48. Number of computations and communications per simulation instant for each agent scenario.

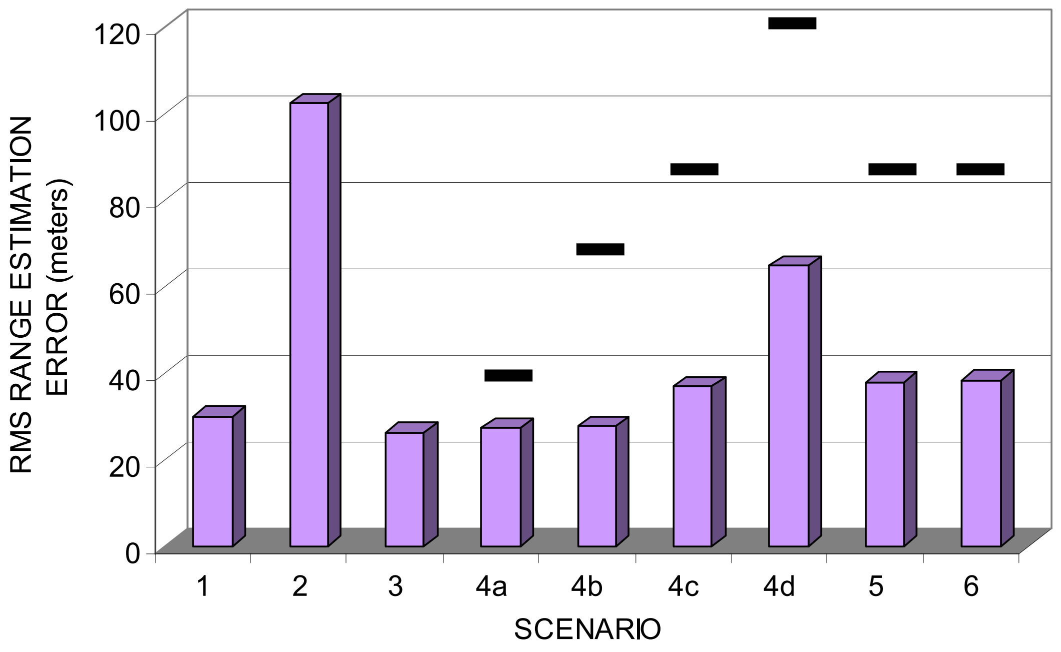


Fig. 49. RMS target range estimation error at the master node for each agent scenario. The horizontal lines indicate $\sqrt{Th}$ set in CCT for scenarios 4-6.

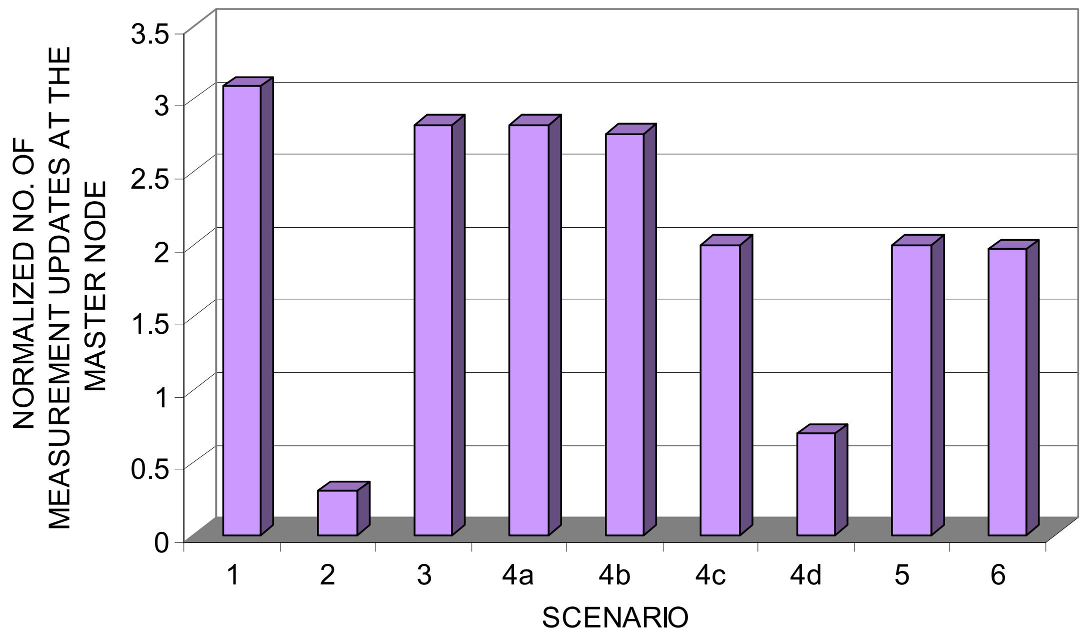


Fig. 50. Number of measurement updates per simulation instant at the master node for each agent scenario.

C. Discussion of Comparison Results for Single Trial Scenario Simulations

Only one master node was present in the system in scenario 1. There were no agents present in the system. No resource management strategies were used. The tracking performance was good (Fig. 49), but because of the large communication overhead (Fig. 48), the simulation ended due to the loss of the only master node (Fig. 21c) at simulation instant 517 (Fig. 22). Since there were no resource-balancing routing schemes implemented in scenario 1, the same sensor nodes were used for routing data over the entire simulation duration. Even though the simulation stopped after only 517 simulation instants, only 12% battery charge was remaining in one of the routing sensor nodes (Fig. 47). This minimum battery charge in the routing sensor node indicates that even if scenario 1 had not stopped at simulation instant 517 due to the loss of the master node, the field in scenario 1 would not be able to last much longer due to the impending loss of the routing sensor node.

Similar to scenario 1, only one master node was present in the system in scenario 2. In scenario 2, TT resulted in intermittent measurements being sent from the cluster nodes to the master node, which resulted in fewer computations and communications taking place in the system (Fig. 48); therefore, more master and cluster battery charge remained at the end of simulations in scenario 2 than in scenario 1 (Fig. 47). Fewer communications, especially between the cluster and the master nodes, reduced the use of the routing sensor nodes, and the minimum battery charge remaining in the sensor nodes in scenario 2 was much larger than for scenario 1 (Fig. 47). Because of TT, fewer measurement updates were performed in scenario 2 (Fig. 50) than in scenario 1, and the tracking performance (Fig. 49) suffered. The TT parameters used in this research might be useful in cases where gradual degradation of the tracking performance is acceptable, but the TT parameters could be modified to achieve a better tracking performance at the expense of battery utilization.

Only one master node was present in scenarios 1 and 2. Multiple master nodes were present in scenarios 3-6. Unlike scenario 2, no adaptive sampling schemes were implemented in scenario 3. In scenario 3, four master nodes were present, and scheme CC1 was used effectively to switch the master node functions from one of the master nodes to another. Scenario 3 lasted for the entire simulation duration of 1100 simulation instants (Fig. 28) compared to scenario 1, which lasted for 517 simulation instants (Fig. 22). Since the system performed measurement updates at every simulation instant, the battery consumption of the nodes (Fig. 27), as well as the trace of the master node *P(k)* (Fig. 29), were similar to scenario 1, until simulation instant 517. Scenario 3 also implemented the localized optimization routing scheme CC4. Scheme CC4 rotated the data routing function among different sensor nodes to balance the battery utilization in the sensor nodes. The minimum sensor node battery charge remaining in scenario 3 was larger than that in scenario 1 (Fig. 47).

Four master nodes were present in scenarios 4a-4d. Scenarios 4a-4d used scheme CC1 effectively to switch the master node functions from one master node to another. Scenarios 4a-4d used scheme CCT to save battery charge (Fig. 46) while attempting to maintain tracking performance (Fig. 49) at a desired threshold. Scenarios 4a-4d were able to achieve a RMS target range estimation error less than $\sqrt{Th}$ (Fig. 49). A linear relationship exists between the RMS target range estimation error and $\sqrt{Th}$ for scenarios 4b-4d (Fig. 49). In case of scenario 4a, because of the low *Th*, measurement updates were performed at every simulation instant, and CCT was unable to influence the RMS target range estimation error. Scenarios 4c and 4d have a large desired tracking performance threshold (Table IX), resulting in fewer computations and communications (Fig. 48), thus helping to conserve battery charge at the master and cluster nodes (Fig. 46). Fig. 51 shows the estimated target track using CCT with the *Th* set to 4230 and 14100. Fig. 52 shows only a selected region of Fig. 51 for better visibility. For a *Th* of 14100 in scenario 4d, the master node performed measurement updates less frequently compared to the *Th* of 4230 in scenario 4b (Fig. 50). Fewer measurement updates resulted in higher tracking errors for high *Th*, especially when the target changed directions (Fig. 52).

In scenarios 5 and 6, schemes CC1, CC2, CC3, and CC4 were implemented along with CCT. The *Th* was set at 7050, as it was set for scenario 4c, resulting in similar tracking performances in scenarios 4c, 5, and 6 (Fig. 49). In scenarios 5 and 6, the number of communications was higher than in scenario 4c (Fig. 48), due to the implementation of schemes CC2 and CC3, which had additional communications overhead. The number of communications in scenario 6 was higher than scenario 5 due to the overhead involved in moving the master node functions among the nine master/cluster nodes.

The primary objective of this dissertation was to present and compare alternative agent paradigms applicable for resource management in sensor networks. Simulation results verify the operation of the different agent-based adaptive sampling and control and coordination algorithms used for resource management. Field layout, number of nodes in the field, initial battery charge in the different nodes, battery-draining weights for different activities, uncertainty in the knowledge of the sensor node positions, and target track are some of the parameters that influence the simulation results. The performance results of the individual scenarios might vary with different choices made for these parameters. However, the fixed set of parameters used in this dissertation established the effectiveness of using agents for resource management in different scenarios.

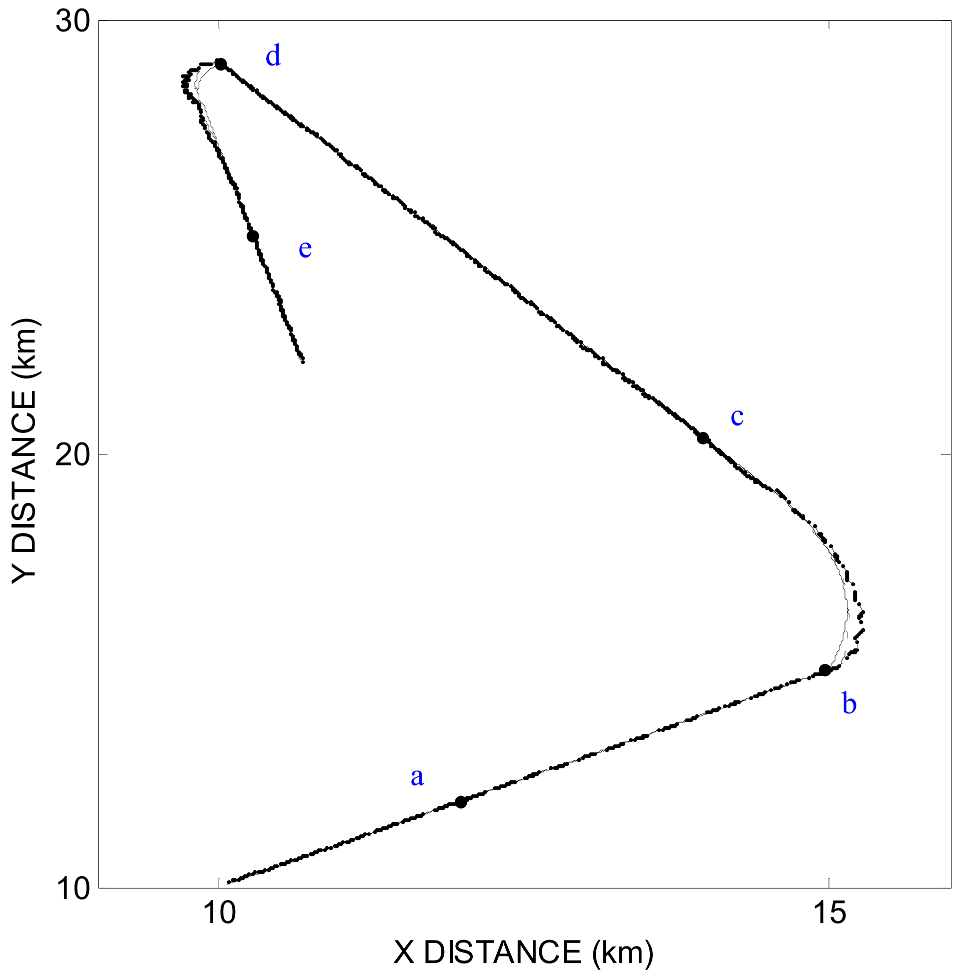


Fig. 51. Representative target track for single-trial simulations (solid line) and estimated target position using CCT with the *Th* set to 4230 (dash-dot line) for scenario 4b, and the *Th* set to 14100 (dotted line) for scenario 4d. Points a, b, c, d, and e indicate the target positions at simulation instants of 100, 250, 400, 830, and 1000, respectively.

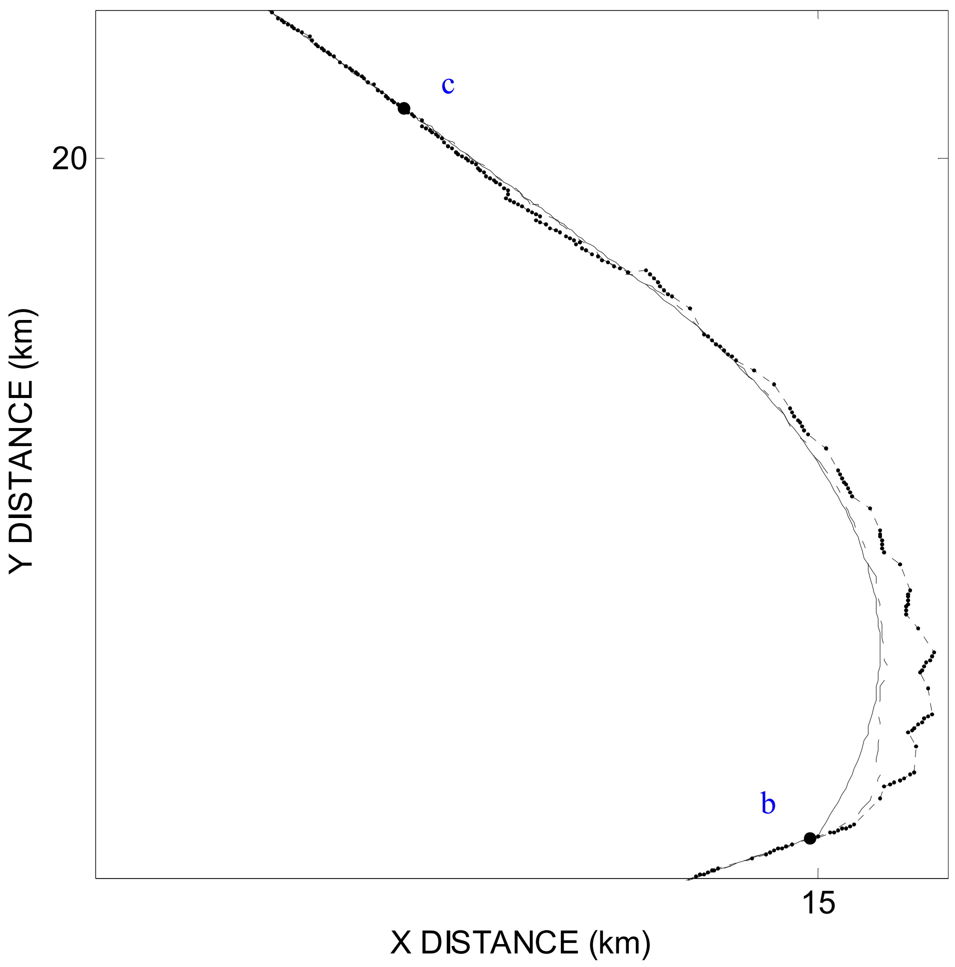


Fig. 52. Selected region of Fig. 51 expanded for better visibility. Estimated target position using CCT with the *Th* set to 4230 (dash-dot line) for scenario 4b, and the *Th* set to 14100 (dotted line) for scenario 4d. Points b and c indicate the target position at simulation instants of 250 and 400.

The MIQs evaluated using (32) for the different agent-based scenarios, the RMS range estimation errors, and the average and minimum battery charge remaining at the end of simulations in the sensor, cluster, and master nodes are presented in Table X. The number of agents in the system is an important factor in determining the MIQ for agent-based sensor network systems. Scenario 1 did not have any agents for resource management in the system. The MIQ of scenario 1 was the BSI of the sensor network. Scenarios 2-6 employed intelligent agents for improved resource management in the network. The single master agent in scenario 2 was able to manage resources at the expense of the tracking performance. Four master agents and nine cluster agents were present in scenario 3, hence scenario 3 had a higher MIQ and was better able to manage resources compared to scenarios 1 and 2.

Scenarios 4a-4d had the same number of agents implementing the same resource management schemes and had the same MIQ. In spite of having the same MIQ, scenarios 4a-4d exhibited different performances as indicated by the tracking error and battery charge remaining in the different nodes in the network (Table X). This performance difference can be attributed to the different *Th* set in scenarios 4a-4d. Scenario 5 had a higher MIQ compared to scenarios 4a-4d and was able to better manage and use resources in the network. Unlike scenarios 1-5, scenario 6 did not have any special master node hardware. In spite of having fewer resources, the stacked functional agents in scenario 6 were able to manage resources quite efficiently. The MIQ did not guarantee any specific level of system performance (Table X). Along with the MIQ, simulation parameters and operational constraints play an important role in determining the system performance.

TABLE X
MIQ FOR DIFFERENT AGENT-BASED SCENARIOS

	SCENARIO								
	1	2	3	4a	4b	4c	4d	5	6
MIQ	25	67	763	999	999	999	999	1868	2673
RMS range estimation error in meters	30	102	26	27	27	36	64	38	38
Average percentage battery charge remaining in the sensor nodes at the end of simulation	93	87	86	86	86	86	87	92	90
Average percentage battery charge remaining in the cluster nodes at the end of simulation	82	90	71	71	71	77	87	82	64
Average percentage battery charge remaining in the master nodes at the end of simulation	0	52	42	42	43	56	82	56	-
Minimum percentage battery charge remaining in the sensor nodes at the end of simulation	13	72	37	37	38	41	59	41	45
Minimum percentage battery charge remaining in the cluster nodes at the end of simulation	59	86	28	28	30	48	76	48	40
Minimum percentage battery charge remaining in the master nodes at the end of simulation	0	52	0	0	0	0	28	0	-

D. Monte Carlo Simulation Results

40 Monte Carlo trials were conducted for scenarios 5 and 6, and the results are presented in Table XI. The average field life at the end of 40 Monte Carlo trials was 2540 simulation instants for scenario 5 and 2647 simulation instants for scenario 6. For each trial, the simulation stopped due to the loss of a sensor or cluster node or due to the loss of all the master nodes in the network. The results from Table XI indicate that during the 40 trials for scenario 5, the simulation ended 34 times due to loss of master nodes and six times due to loss of cluster nodes. In scenario 6, the simulations ended due to the loss of the master/cluster node for all 40 trials. The field life achieved in scenario 5 could be increased by increasing the number of master nodes or by increasing the battery capacities of the available master nodes. The field life achieved in scenario 6 could be increased by increasing the battery capacities of the available master/cluster nodes. Though scenario 6 had a much larger MIQ than scenario 5 (Table X), comparable field lives were achieved for scenarios 5 and 6 during the 40 Monte Carlo trials (Table XI). The simulation parameters and operational challenges in scenario 6 thus did not allow the capabilities of the system to be fully used in order to achieve a higher field life than scenario 5.

TABLE XI
SIMULATION RESULTS FOR SCENARIO 5 AND 6 IN
40 MONTE CARLO TRIALS

	SCENARIO	
	5	6
Average field life (simulation instants)	2540	2647
Standard deviation field life (simulation instants)	480	578
Average number of computations per simulation instant	2	2
Average number of communications per simulation instant	6	7
Number of times simulation ended due to loss of master nodes	34	-
Number of time simulation ended due to loss of cluster nodes	6	30
Average percentage battery charge remaining in the sensor nodes at the end of simulation	83	80
Average percentage battery charge remaining in the cluster nodes at the end of simulation	65	23
Average percentage battery charge remaining in the master nodes at the end of simulation	0.17	-

E. Summary

Single-trial simulation results for scenarios 1-6 were presented and compared in this chapter. For scenarios 5 and 6, 40 Monte Carlo simulation trials were performed, and the simulation results were presented. Scenarios 5 and 6, in which multiple master or master/cluster nodes were present, measurement updates were performed intermittently using CCT, and clusters and sensors far away from the target were switched off, were able to attain a good balance of tracking performance and field life.

VIII. CONCLUSION

Agent-based systems are a new and robust paradigm for designing flexible architectures for systems with disparate needs. Agents have much to offer in sensor networks, but since this is an emerging field, the literature offering guiding principles or strategies for using agents in sensor networks is sparse. The results presented in this dissertation provide an insight into the use of agents in DSNs and may assist designers of agent-based systems in using agents to their best advantages in scenarios similar to the wide range of those explored.

In this dissertation, different agent paradigms developed for an undersea distributed sensor network were explored and compared. The use of agents to achieve the goals of managing resources to increase field life in DSNs was demonstrated. Even though a particular undersea application was considered, many of the results from this research can be applicable to other agent-based systems.

A modular simulation framework based on object-oriented design was used to generate data for a detailed analysis of component interactions and for the performance evaluation of the different agent-based scenarios. The simulation framework demonstrated the scalability and flexibility of agent-based systems where task-specific agents are added or removed to simulate different scenarios with ease. Agents were able to balance the tracking performance and survivability in the DSN with different adaptive sampling and control and coordination schemes.

Adaptive sampling schemes CCT and TT were implemented by the agents to balance tracking performance and resource utilization in the network. The TT parameters used in this research might be useful in cases where gradual degradation of tracking performance is acceptable in order to conserve battery resources. The TT parameters could be modified to achieve a better tracking performance at the expense of battery utilization. CCT, on the other hand, is useful for maintaining the tracking performance at an acceptable level while still balancing the network resources.

Scheme CC1 was useful in balancing network resources by rotating the master node functions among multiple master nodes. Schemes CC2 and CC3 helped to conserve battery charges in the cluster and sensor nodes by turning off nodes that were far away from the target position. Scheme CC4 used localized optimization routing to balance power consumption in the sensor nodes by rotating the routing function among different sensor nodes.

A measure of the machine intelligence quotient for agent-based systems was developed to facilitate comparison of different agent paradigms. The MIQ gives a theoretical measure for quantifying the benefits of the decisions made by the agents to achieve the goals in the sensor network. Scenarios having high MIQs generally achieved a high performance, though simulation parameters and operational challenges sometimes may not have allowed the capabilities of the system to be fully used. The MIQ measure developed in this dissertation for comparing agent-based systems can be helpful in making design and development decisions for other agent-based applications.

Future work might involve adjusting the threshold in CCT frequently in order to achieve improved tracking performance. A combination of TT and CCT might also be explored to balance field life and tracking performance. Instead of using sensor nodes that measure the range and bearing of the target, research in agent-based sensor networks using binary sensor nodes that detect only the presence or absence of the target could also be considered.

REFERENCES

[1] SmartMesh-XD. [Online]. Available: http://www.dustnetworks.com/products/smartmesh-xd.shtml, [21 June, 2007].

[2] Intel Mote. [Online]. Available: http://www.intel.com/research/exploratory/motes.htm, [21 June, 2007].

[3] Crossbow Wireless Sensor Networks. [Online]. Available: http://www.xbow.com/Products/productdetails.aspx?sid=156, [21 June, 2007].

[4] Moteiv Tmote Sky. [Online]. Available: http://www.moteiv.com/products/tmotesky.php, [21 June, 2007].

[5] Wireless Ad Hoc Sensor Networks. [Online]. Available: http://w3.antd.nist.gov/wahn_ssn.shtml, [21 June, 2007].

[6] N. Xu, A survey of sensor network applications. [Online]. Available: http://enl.usc.edu/~ningxu/papers/survey.pdf, [May 19, 2005].

[7] C. C. Hayes, "Agents in a nutshell - a very brief introduction," *IEEE Trans. Knowl. Data Eng.,* vol. 11, no. 1, pp. 127-133, Jan. 1999.

[8] M. Tubaishat and S. Madria, "Sensor networks: an overview " *IEEE Potentials,* vol. 22, no. 2, pp. 20-23, Apr. 2003.

[9] D. Culler, D. Estrin, and M.B. Srivastava, "Overview of sensor networks," *IEEE Computer,* vol. 37, no. 8, pp. 41-49, Aug. 2004.

[10] C. Y. Chong, S. P. Kumar, and B. A. Hamilton, "Sensor networks: evolution, opportunities, and challenges," *Proc. IEEE,* vol. 91, no. 8, pp. 1247-1256, Aug. 2003.

[11] A. Mainwaring, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *ACM Int. Workshop on Wireless Sensor Networks and Applications (WSNA),* 2002.

[12] N. Kurata, B. F. Spencer, M. Ruiz-Sandoval, Y. Miyamoto, and Y. Sako, "A study on building risk monitoring using wireless sensor network MICA-mote," in *Proc. 1st Int. Conf. Structural Health Monitoring and Intelligent Infrastructure,* 2003, pp. 353-357

[13] V. A. Kottapalli, A. S. Kiremidjian, J. P. Lynch, E. Carryer, T. W. Kenny, K. H. Law, and Y. Lei, "Two-tiered wireless sensor network architecture for structural health monitoring," in *Proc. 10th Annu. Int. Symp. Smart Structures and Materials*, San Diego, CA, 2003, pp. 8-19.

[14] C. L. Fok, G. C. Roman, and C. Lu, "Mobile agent middleware for sensor networks: An application case study," in *Proc. 4th Int. Conf. Inform. Process. Sensor Networks (IPSN)*, 2005, pp. 382 -387.

[15] S. McGirr, K. Raysin, C. Ivancic, and C. Alspaugh, "Simulation of underwater sensor networks," *OCEANS MTS/IEEE. Riding the Crest into the 21st Century,* vol. 2, pp. 945-950, Sept. 1999.

[16] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor network," *Proc. IEEE,* vol. 91, no. 8, pp. 1163-1171, Aug. 2003.

[17] T. Stevens and D. Morrell, "Minimization of sensor usage for target tracking in a network of irregularly spaced sensors," in *IEEE Workshop Statistical Signal Process.,* Sept. 2003, pp. 549-552.

[18] J. P. Wagner and R. Cristescu, "Power control for target tracking in sensor networks," in *Proc. 39th Conf. Inform. Sciences and Systems (CISS),* Baltimore, MD, Mar. 2005.

[19] Y. Xue and D. Morrell, "Adaptive foveal sensor for target tracking," in *36th Asilomar Conf. Signals, Syst., and Comput.,* Nov. 2002, pp. 848-852.

[20] X. Yu, K. Niyogi, S. Mehrotra, and N. Venkatasubramanian, "Adaptive target tracking in sensor networks," in *The 2004 Communication Networks and Distributed Syst. Modeling and Simulation Conf. (CNDS),* San Diego, CA.

[21] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks,* vol. 3, no. 3, pp. 257-279, May 2005.

[22] J. Hei, Y. Li, A. Syed, J. Wills, and W. Ye, "Research challenge and applications for underwater sensor networking," in *Proc. IEEE Wireless Commun. and Networking Conf. (WCNC)*, 2006, pp. 228-235.

[23] H. Qi, P. T. Kuruganti, and Y. Xu, "The development of localized algorithms in wireless sensor network," *Sensors,* vol. 2, no. 7, pp. 286-293, Jul. 2002.

[24] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *Proc. IEEE Workshop Sensor Network Protocols and Applications*, May 2003, pp. 71-81.

[25] R. S. Bowyer and R. E. Bogner, "Cooperative behaviour in multi-sensor systems," in *Proc. 6th Int. Conf. Neural Inform. Process.,* Nov. 1999, pp. 807-812.

[26] N. Patwari and A. O. Hero, "Hierarchical censoring for distributed detection in wireless sensor networks," in *IEEE Int. Conf. Acoustics, Speech, and Signal Process. (ICASSP),* Apr. 2003, pp. 848-851.

[27] X. Du and F. Lin, "Efficient energy management protocol for target tracking sensor networks," in *9th IFIP/IEEE Int. Symp. Integrated Network Management*, 2005, pp. 45-58.

[28] A. Jain and E. Y. Chang, "Adaptive sampling for sensor networks," in *Proc. 1st Int. Workshop Data Management for Sensor Networks: in Conjunction with VLDB*, 2004, pp. 10-16.

[29] R. Willett, A. Martin, and R. Nowak, "Backcasting: adaptive sampling for sensor networks," in *Proc. 3rd Int. Symp. Inform. Process. Sensor Networks*, 2004, pp. 124-133.

[30] J. L. Wong, S. Megerian, and M. Potkonjak, "Staggered sampling for energy efficient data collection," in *Proc. 5th IEEE Conf. Sensors*, 2007, pp. 777-780.

[31] J. Kho, A. Rogers, and N. R. Jennings, "Decentralised adaptive sampling of wireless sensor networks," in *Proc. 1st Int. Workshop Agent Technology Sensor Networks, a Workshop 6th AAMAS*, 2007, pp. 55-62.

[32] M. Kalandros and L. Y. Pao, "Multisensor covariance control strategies for reducing bias effects in interacting target scenarios," in *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 1, pp. 153-173, Jan. 2005.

[33] M. Wooldridge and N. R. Jennigs, "Intelligent agents: Theory and practice," *The Knowledge Engineering Review,* vol. 10, no. 2, pp. 115-152, Jan. 1995.

[34] T. S. Perraju, "Agents and autonomous distributed systems," *Proc. 4th Int. Symp. Autonomous Decentralized Syst.,* Mar. 1999, pp. 264-266.

[35] V. Marik and D. McFarlane, "Industrial adoption of agent-based technologies," *IEEE Intell. Syst.,* vol. 20, no. 1, pp. 27-35, Jan. 2005.

[36] T. Schetter, M. Campbell, and D. Surka, "Multiple agent-based autonomy for satellite constellations," *J. Artificial Intell.,* vol. 145, no. 1, pp. 147-180, Apr. 2003.

[37] T. S. Perraju, "An agent framework for survivable network systems," in *IEEE Int. Performance, Computing and Communications Conf.,* Feb. 1999, pp. 469-475.

[38] H. S. Nwana, "Software agents: an overview," *Knowledge Engineering Review,* vol. 11, no. 3, pp. 1-40, Sept. 1996.

[39] E. A. Kendall, M. T. Malkoun, and C. H. Jiang, "The application of object-oriented analysis to agent based systems," *The J. of Object Oriented Programming,* vol. 5, pp. 45-52, Feb. 1997.

[40] M. Wooldridge and N. R. Jennings, "Application of intelligent agents," in *Agent Technology: Foundations, Applications, and Markets*, New York, NY: Springer-Verlag Inc., 1998, pp. 3-28.

[41] IEEE/WIC/ACM Int. Conf. Intelligent Agent Technology, IAT 06 in Hong Kong [Online]. Available: http://www.comp.hkbu.edu.hk/iwi06/iat/ [Dec. 6, 2006].

[42] 6th Int. Joint Conf. Autonomous Agents and Multiagent Syst. (AAMAS 07) in Hawaii [Online]. Available: http://www.ecs.soton.ac.uk/~acr/atsn/ [Dec. 6, 2006].

[43] Intelligent Mobile Agents in Wireless Sensor Networks. [Online]. Available: http://www.cs.wustl.edu/mobilab/projects/agilla/, [May 19, 2005].

[44] D. A. Roozemond and J. L. H. Rogier, "Agent controlled traffic lights," in *European Symp. Intelligent Techniques (ESIT)*, 2000, pp. 78-82.

[45] D. D. Corkill, D. Holzhauer, and W. Koziarz, "Turn off your radios! Environmental monitoring using power-constrained sensor agents," in *1st Int. Workshop Agent Technology for Sensor Networks (ATSN)*, Honolulu, HI, 2007, pp. 31-38.

[46] L. Andersson and A. Rönnbom, "Intelligent agents – a new technology for future distributed sensor systems," M.S. thesis, Dept. Informatics, Göteborg Univ., 1999.

[47] H. Q. X. Wang, S. S. Iyengar, and K. Chakrabarty, "Multi-resolution data integration using mobile agents in distributed sensor networks," *IEEE Trans. Syst. Man, and Cybern.,* vol. 31, no. 3, pp. 383-391, Aug. 2001.

[48] H. Qi, X. Wang, S. S. Iyengar, and K. Chakrabarty, "High performance sensor integration in distributed sensor networks using mobile agents," *Int. J. High Performance Computing Applications,* vol. 16, no. 3, pp. 325-335, Mar. 2002.

[49] J. J. Marciniak, *Encyclopedia of Software Engineering*. New York, NY: John Wiley & Sons, 1994.

[50] W. Harrison, "An entropy-based measure of software complexity," *IEEE Trans. Softw. Eng.,* vol. 18, no. 11, pp. 1025-1029, Nov. 1992.

[51] R. R. Brooks, "An intelligence metric for emergent distributed behaviors," in *NIST Workshop on Performance Metrics for Intell. Syst.* Gaithersburg, MD, 2000.

[52] J. Albus, "Outline for a theory of intelligence," *IEEE Trans. on Systems, Man, and Cybernetics,* vol. 21, no. 3, pp. 473-509, Jun. 1991.

[53] R. Gao and L. H. Tsoukalas, "Performance metrics for intelligent systems: An engineering perspective," in *NIST Workshop on Performance Metrics for Intell. Syst.*, Gaithersburg, MD, 2002.

[54] R. Simmons, "Survivability and competence as measures of intelligent systems," in *NIST Workshop on Performance Metrics for Intell. Syst.*, Gaithersburg, MD, 2002.

[55] I. R. Nourbakhsh, "Two measures for the "intelligence" of human-interactive robots in contests and in the real world: expressiveness and perceptiveness," in *NIST Workshop on Performance Metrics for Intell. Syst.*, Gaithersburg, MD, 2002.

[56] A. Yavnai, "Metrics for system autonomy," in *NIST Workshop on Performance Metrics for Intell. Syst.,* Gaithersburg, MD, 2002.

[57] S. Lee, W. Bang, and Z. Z. Bie, "Measure of system intelligence: An engineering prespective," in *NIST Workshop on Performance Metrics for Intell. Syst.,* Gaithersburg, MD, 2000.

[58] H. J. Park, B. K. Kim, and K. Y. Lim, "Measuring the machine intelligence quotient (MIQ) of human-machine cooperative systems," *IEEE Trans. Syst. Man, Cybern. A., Syst. Humans,* vol. 31, no. 2, pp. 89-96, Mar. 2001.

[59] S. V. Chandrachood, "Investigation of coordination between sensor network using resource based models," M.S. thesis, Dept. Elect. and Comput. Eng., The Univ. Alabama, Birmingham, 2005.

[60] S. V. Chandrachood, A. Anthony, and T. C. Jannett, "Using resource based modeling to evaluate coordination schemes in wireless sensor networks," in *Proc. IEEE SoutheastCon*, Memphis, TN, 2006, pp. 91-97.

[61] R. Praveenkumar, "Investigating routing protocols in sensor networks," M.S. thesis, Dept. Elect. and Comput. Eng., The Univ. Alabama, Birmingham, 2006.

[62] R. Praveenkumar and T. C. Jannett, " Investigation of routing protocols in a sensor network using resource based models," in *Proc. IEEE SoutheastCon*, Richmond, VA, 2007, pp. 29-34.

[63] J. Manyika and H. Durrant-White, *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Upper Saddle River, NJ: Prentice Hall, 1994.

[64] A. G. O. Mutambara, *Decentralized Estimation and Control for Multi-Sensor Systems*. Boca Raton, FL: CRC Press, 1998.

[65] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME - J. Basic Eng.,* pp. 35-45, Mar. 1960.

[66] G. Welch and G. Bishop, An Introduction to the Kalman Filter. [Online]. Available: http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf, [21 June, 2007]

[67] R. Kasthurirangan, "Comparison of architectures for multi-sensor data fusion in deployable autonomous distributed systems," M.S. thesis, Dept. Elect. and Comput. Eng., The Univ. Alabama, Birmingham, 2003.

[68] J. Utts and R.F. Heckard, *Mind on Statistics*. Pacific Grove, CA: Duxbury Press, 2006, ch. 9, pp. 358-362.