Resource Management in Wireless Sensor Networks Hosting Multiple Applications

by

Navdeep Kaur Kapoor, M.A.Sc (ECE), B.E (EE)

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

Your file Votre référence
ISBN: 978-0-494-94224-6

Our file Notre référence
ISBN: 978-0-494-94224-6

Abstract

Sensor grids comprise of wireless sensor networks (WSNs) and the computing grid. In a sensor grid real time information about a phenomenon being monitored can be obtained from a wireless sensor network and processed in a computing grid. With the advancement of sensor technology and with the aim to increase the cost efficiency of wireless sensor networks, multi-purpose wireless sensor networks are gaining popularity over wireless sensor networks dedicated to a single application. When a WSN is shared by more than one application, management of sensor resources becomes a challenging task. Resource management in a WSN comprises of two major functions: Allocation and Scheduling. This research presents a simulation based investigation of allocation and scheduling on wireless sensor networks hosting multiple applications. This thesis proposes scheduling algorithms that combine the knowledge of the application and the system in order to make a scheduling decision. Simulation experiments demonstrate that such algorithms provide a better performance than the knowledge free First Come First Served (FCFS) algorithm. Simulation experiments demonstrate that communication delays play a significant role in determining the overall mean response time in a WSN and a scheduling algorithm that allocates a higher weight to higher hop distances from the cluster head provides the best performance.

A number of allocation techniques are also proposed in this thesis. The aim of any allocation algorithm is to allocate sensors to application requests in order to enhance the network lifetime of the wireless sensor network. Allocation algorithms that use varying degrees of information about the state of the sensor nodes and the network are proposed

in this research. Both static and dynamic allocation techniques are proposed in this research. Experiments demonstrate that by using information about the energy consumption at the CPU component and the radio component of the sensor nodes, and making dynamic allocation of sensor nodes to application requests in the wireless sensor network, lifetime of the WSN can be increased.

Acknowledgments

First and foremost, I would like to express my sincere thanks to my thesis advisors Prof. Shikharesh Majumdar and Prof. Biswajit Nandy for their continuous support, encouragement and help throughout the thesis. I am thankful to Natural Sciences and Engineering Research Council of Canada, Government of Ontario and Carleton University for providing financial support for this research. I am grateful to all reviewers whose comments helped me to shape this thesis.

I would like to express my deep appreciation and thanks to my husband Balwinderjit, for his patience, understanding and strong support. Without his support it would not have been possible to reach this milestone.

Last but not the least, I want to thank my kids, Arshia and Ankit, who could not get the time, they deserved and needed the most due to my involvement in the research.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

λ	Arrival Rate of Requests
λ_{appi}	Arrival Rate of Requests for an Application
E_{appi}	Execution Time of an Application
R_t	Mean Response Time
N_s	Total Number of Sensors
$N_{applications}$	Number of Applications
p	Probability of Request Arrival
S_{appi}	Number of Sensors Required by an Application
API	Application Programming Interface
BMA	Balanced Metric Allocation
CLBA	CPU Load Balanced Allocation
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Access
CTS	Clear To Send
DCLBA	Dynamic CPU Load Balanced Allocation
DDLBA	Dynamic Data Load Balanced Allocation
DLBA	Data Load Balanced Allocation
DRA	Dynamic Random Allocation
EAR	Eavesdrop-And-Register
EDF	Earliest Deadline First

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

FCFS	First Come First Served
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FNDP	Farthest Number Distance Product
GPSR	Greedy Perimeter Stateless Routing
LEACH	Low-energy Adaptive Clustering Hierarchy
LFNDPF	Least Farthest Number Distance Product First
LLF	Least Laxity First
LNDPF	Least Number Distance Product First
LNHF	Least Number of Hops First
LNSF	Least Number of Sensors First
LWFNDPF	Least Weighted Farthest Number Distance Product First
MAC	Medium Access Control
MEF	Maximum Energy First
NDP	Number Distance Product
NSF	National Science Foundation
OGSA	Open Grid Services Architecture
QoM	Quality of Monitoring
QoS	Quality of Service
RA	Random Allocation

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

RTS	Ready To Send
RFID	Radio Frequency Identification
SAR	Sequential Assignment Routing
SJN	Shortest Job Next
SMACS	Self-Organizing Medium Access Control for Sensor Networks
SMECN	Small Minimum Energy Communication Network
SPIN	Sensor Protocols for Information via Negotiation
SPRING	Scalable Proxy-based Architecture for Sensor Grid
SQDDP	Sensor Query and Data Dissemination Protocol
SQTL	Sensor Query and Tasking Language
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
UMADE	Utility-based Multi-application Allocation and Deployment Environment
VSN	Virtual Sensor Network
WFNDP	Weighted Farthest Number Distance Product
WSN	Wireless Sensor Network
XML	Extensible Markup Language

Chapter 1: Introduction

1.1 Background

Grid Computing emerged as a paradigm for distributed computing. It promotes sharing of distributed resources that may be heterogeneous in nature and enables scientists and engineering professionals to solve large scale computing problems. With grid computing, businesses and institutions can efficiently utilize computing and data resources and combine them for large capacity workloads. The primary benefit of grid computing is the ability to coordinate and share resources. Today, use of grid computing can be seen in larger businesses, for resource demanding applications and department specific processes [1]. Ian Foster, head of the Distributed Systems Laboratory at Argonne National Laboratory, has given a simple three-point checklist to define a grid [2]. He has defined a grid as a system that,

1. "Coordinates resources that are not subject to centralized control."
2. "Uses standard, open, general-purpose protocols and interfaces."
3. "Delivers nontrivial qualities of service."

In other words, grid computing is a way to share computing resources within and among organizations. The resources in a grid may include computational resources, storage resources, network resources, and code repositories [1]. These resources are a part of a "virtual organization" [3], which may comprise of several individuals and institutions. The resources in a virtual organization are connected via a network (possibly by a private network or the internet) and may be owned by different organisations or

individuals and are shared under locally defined algorithms. Recently, cloud computing has evolved from grid computing and provides on demand resource provisioning.

In recent times, there has been a lot of advancement in the field of wireless communication. Also the capabilities and processing power of wireless devices has improved significantly in recent years. The concept of a wired grid has been extended to wireless devices and wireless grids have become an extension to the wired grids [4] [5]. Similar to wired grids; wireless grids are distributed in nature. Wireless grids also need to follow common protocols and standards for communication and there is also a need for appropriate security and Quality of Service levels [6].

Wireless grids are typically used in case of the following applications [4] [7]:

- Applications that collect and aggregate data;
- Applications that take advantage of the locations at which they can exist or can move to where it is difficult for their wired counterparts to exist;
- Applications based on the cooperation among a mesh of mobile wireless devices.

Various classes of wireless grids are discussed in the next section.

1.2 Types of Wireless Grids

Wireless grids have been characterized based on the characteristics of the devices comprising the grid and the relative mobility of the devices comprising the grid [8].

1.2.1 Sensor Networks and Grids

Sensor grids are the integration of two technologies: wireless sensor networks and the grid. Real time information about a phenomenon being monitored can be obtained from a wireless sensor network and can be processed in a computing grid [9]. Sensor networks are composed of tiny devices (sensor nodes) comprising of a sensor or multiple sensors, a battery, and a radio transmitter. The term sensor and sensor node are used interchangeably in literature. The purpose of sensor nodes is to cover a particular geographic region with the aim of detecting a particular phenomenon. The detected information may be processed by the sensors and then communicated to a sink via other sensors based on a routing algorithm. The sink finally transmits this information to the computing devices in the grid. Although a sensor grid comprises of a sensor network interfaced with a computing grid, the terms sensor networks and sensor grids are used interchangeably in literature. Wireless sensor networks (WSNs) integrate detection, processing, and communication into the grid [10]. Presently, sensors are used in applications that may involve measurement of temperature, pressure, humidity, movement detection, radiation, and particulate levels (level of particles suspended in a gas or liquid) [4]. Similar to sensor grids which have evolved from computing grid, sensor clouds have also evolved from cloud computing. Sensors provide information in real time to the cloud and the immense processing power of the cloud is used to provide quick response to the users [11] [12].

1.2.2 Fixed Wireless Grids

In case of a fixed wireless grid, the topology of the grid remains fixed similar to the wired grid. The wireless devices are not mobile and do not enter or leave the grid. The challenges faced by a fixed wireless grid are similar to the challenges faced by a wired grid except for the additional challenges faced due to wireless communication [8]. The most commonly seen example of a fixed wireless grid is a home network of computers with wired connectivity replaced by wireless connections [8] [13].

1.2.3 Mobile/Dynamic Wireless Grids

In case of dynamic wireless grids or mobile wireless grids, the devices are mobile and may frequently enter or leave the grid. The configuration of such a grid keeps changing over time. In recent times, the capabilities of mobile devices have increased significantly. These devices connect to the grid through adhoc connections and use the resources available in the grid, and may also make themselves available for use by other members of the virtual organization corresponding to the grid [4].

1.3 Motivations for the Thesis

The research in this thesis is in the area of sensor grids. Most of the research in literature considers dedicated wireless sensor networks, in which the wireless sensor network serves a single application. The primary reason for having a dedicated wireless sensor network is its simplicity. However, a wireless sensor network dedicated to a single application may not be very cost effective. To increase the efficiency of the wireless sensor network, it is often recommended that a wireless sensor network serve multiple applications at the same time [14]. Sharing of wireless sensor networks by multiple

applications has started receiving significant attention from researchers [15] [16] [17] [18]. Some researchers rightfully suggest that no single application would warrant widespread deployment of a WSN and a WSN will be more effectively utilized by multiple diverse applications [14]. When a wireless sensor network supports multiple applications, management of resource constrained sensors becomes even more challenging. The research presented in this thesis concerns the resource management issues in a wireless sensor network hosting multiple applications. A virtual sensor network may be formed from a subset of sensors in a wireless sensor network serving an application at a given time. The sensor nodes comprising a virtual sensor network may belong to the same or different administrative domains. The nodes collaborate to complete a certain task and serve a request from users of an application. A wireless sensor network may comprise multiple such virtual sensor networks. Sensor nodes may belong to multiple virtual sensor networks. Virtual sensor networks are useful in case of applications that are deployed in the same geographic area [19]. They are also useful in wireless sensor networks serving multiple applications and comprising of multi-function sensor nodes. The concept of virtual sensor networks is depicted in Figure 1-1. The figure shows three wireless sensor networks that belong to three administrative domains. Sensors from each of these domains collaborate to serve a given task/application forming a virtual sensor network.

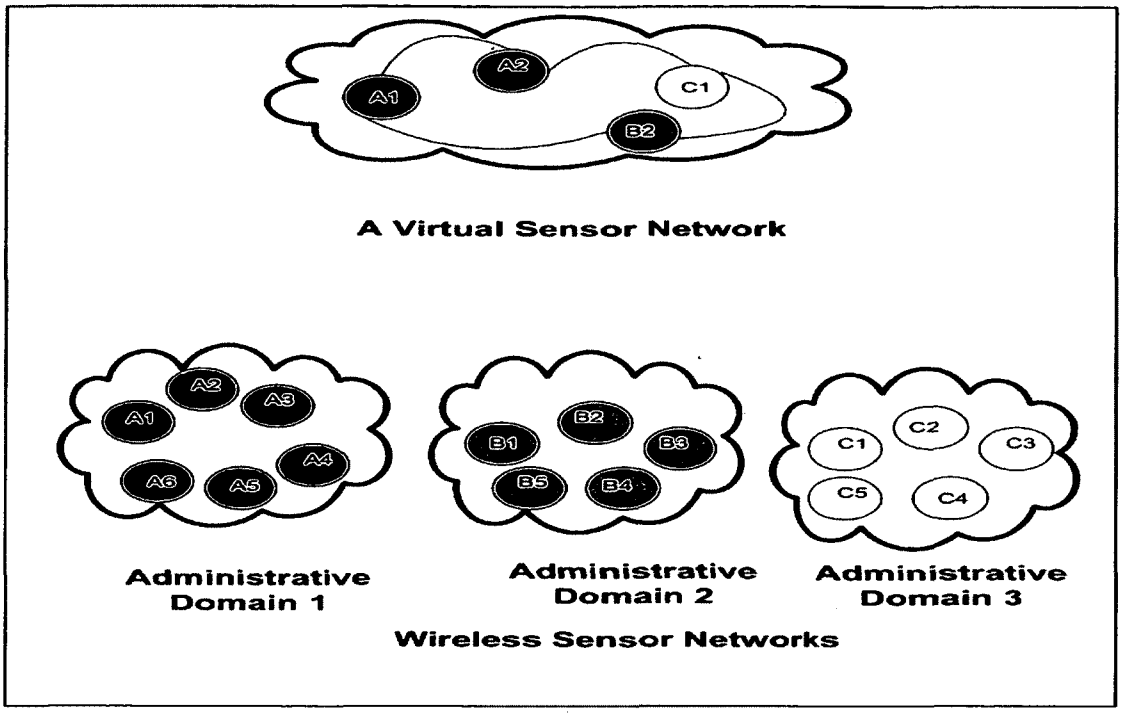


Figure 1-1: A Virtual Sensor Network

With the advancement in technology, sensor nodes may be equipped with sensors having different sensing capabilities. This further promotes sharing of sensors amongst various applications [20]. Sharing of wireless sensor networks has been discussed extensively in literature. Some examples of sharing of wireless sensor networks as discussed in literature are described in Chapter 2.

1.4 Goals of the Thesis

A wireless sensor network comprises of sensor nodes that have limited processing, storage, and power resources. As discussed in Section 1.3, a wireless sensor network can serve more than one application. Different applications may have different attributes in terms of number of sensors required, data size of request and response messages, location of the sensors with respect to the sink, and power consumption.

The management of resource constrained sensor nodes becomes even more challenging when the wireless sensor network is shared by multiple applications. The two most important aspects of resource management in a wireless sensor network are scheduling and allocation. *Scheduling* determines the order in which the requests for the various applications that are submitted by the users should be executed. The order in which the application requests are scheduled has a strong impact on the overall mean response time to the users of applications. *Allocation* refers to the process of selection of sensor nodes from a set of available nodes that will be used to execute a request for an application. The main aim of allocation is to balance the energy consumption amongst the sensor nodes comprising the wireless sensor network. This enhances the lifetime of the individual sensor nodes and hence the lifetime of the network.

The goal of this thesis is to study the problem of resource management in wireless sensor networks hosting multiple applications. The thesis proposes various allocation algorithms and scheduling algorithms for multi-purpose wireless sensor networks. The allocation algorithms primarily aim at improving the network lifespan while the scheduling algorithms aim to improve the mean response time to the users of the application.

1.5 Contributions of the Thesis

The contributions of this thesis are summarized next.

- Effective algorithms for scheduling multiple applications in a wireless sensor network are proposed. This research proposes scheduling algorithms that

combine the knowledge of the application and the system in order to make a scheduling decision.

  - A simulation-based comparative analysis of the performances of scheduling algorithms that use knowledge of application and network with algorithms that do not use such knowledge is performed. Simulation experiments demonstrate that such algorithms provide a better performance than the knowledge free First Come First Served algorithm.
  - The simulation results provide insights into the effect of various system and workload parameters on system performance achieved with the various scheduling algorithms. For example, simulation experiments demonstrate that communication delays play a significant role in determining the response time of applications and hence the overall mean response time in a WSN. The messages to and from the farthest sensors experience large communication delays due to interference from neighboring sensors at each hop. Thus, by using information about the farthest sensors required by each application in scheduling, significant gains can be achieved in system performance.
  - A scheduling algorithm that allocates a higher weight to higher hop distances provides the best performance.

- Algorithms for allocation of job requests belonging to various applications to appropriate sensor nodes are investigated. Both static and dynamic allocation algorithms are proposed. Also, both algorithms based on knowledge of system

and application characteristics as well as algorithms that do not use such knowledge are considered.

  - The dynamic allocation algorithms demonstrate a superior performance over the static allocation algorithms for most of the configurations experimented with.
  - Simulation experiments demonstrate superior performance of an allocation algorithm that uses information about the energy consumption both at the CPU and at the radio component of the sensor nodes.

- Importance of using the knowledge of system and application characteristics in allocation and scheduling is demonstrated. The thesis identifies the system and workload parameters that are important in making resource management decisions.

The following publications have resulted so far from the thesis research.

- Kapoor N., Majumdar S., and Nandy B., "Scheduling on Wireless Sensor Network Hosting Multiple Applications", In the Proceedings of IEEE International Conference on Communications (ICC 2011), Kyoto, Japan, June 2011.

- Kapoor N., Majumdar S., and Nandy B., "System and Application Knowledge Based Scheduling of Multiple Applications in a WSN", In the Proceedings of IEEE International Conference on Communications (ICC 2012), Ottawa, ON, Canada, June 2012.

- Kapoor N., Majumdar S., and Nandy B., "Sensor Allocation to Multiple

Applications in Shared Wireless Sensor Networks", Extended Abstract for Poster, In the Proceedings of 31st IEEE International Performance Computing and Communications Conference (IPCCC 2012), Austin, TX, U.S.A., December 2012.

1.6 Thesis Outline

The thesis comprises of six chapters. The applications, characteristics, and limitations of wireless sensor networks are considered in detail in Chapter 2. Chapter 2 also discusses the problem of resource management in a wireless sensor network. A representative set of works done by other researchers in this field are presented. Chapter 3 provides details of the simulation model, simulation toolkit and the resource management algorithms proposed in this research. It presents the various allocation and scheduling algorithms in detail. It also provides an overview of the system parameters, workload parameters, and performance measures. Chapter 4 provides the results related to the scheduling algorithms proposed in this research. Chapter 5 provides the results related to the allocation algorithms proposed in this research. Chapter 6 concludes by providing a summary of the important findings of this research and provides recommendations for future research.

Chapter 2: Overview of Wireless Sensor Networks

This chapter provides an overview of the area and summarizes a representative set of related works. Some of the potential applications of wireless sensor networks are described in Section 2.1. The applications emphasize the importance of research in the field of wireless sensor networks. Section 2.2 discusses multi-purpose wireless sensor networks. The communication architecture of wireless sensor networks is discussed from a high level perspective in Section 2.3. The design issues and challenges in a wireless sensor network are discussed in detail in Section 2.4. Finally, Section 2.5 discusses resource management in wireless sensor networks and related work.

2.1 Applications of Wireless Sensor Networks

Sensor networks may comprise of different types of sensors: seismic, low magnetic rate, thermal, visual, infrared, and radar. With such a large variety in the types of sensors, several kinds of ambient conditions that include the following may be monitored [21] [22]:

- Temperature
- Humidity
- Movement of Vehicles
- Pressure
- Soil Makeup
- Noise Levels
- Lightning Conditions
- Speed, and direction of movement of an object

- Stress level on attached objects
- Presence, or absence of certain kinds of objects

Various applications of sensor networks have been categorised and are discussed in the following subsections [22]:

2.1.1 Military Applications

Sensor networks have been used in the military for the purpose of monitoring friendly forces, equipment and ammunition, battlefield surveillance, reconnaissance of opposing forces and terrain, targeting, battle damage assessment; and nuclear, biological and chemical (NBC) attack detection and reconnaissance [22].

2.1.2 Environmental Applications

One of the most important applications of wireless sensor networks in the domain of environmental applications is forest fire detection. Sensors may be deployed densely in a forest region and the exact origin of fire may be relayed before the fire becomes uncontrollable. Wireless sensor networks also have numerous applications in the field of agriculture. They may be used to monitor conditions that may affect crops and livestock. They may be used for detection of levels of various chemicals in soil. Movement of birds, animals, and insects may be monitored with the help of sensor networks as well. Sensor networks may also be used for flood detection [23] [24].

2.1.3 Health Applications

Sensor networks have numerous applications in the area of health monitoring. Sensor networks may be used for telemonitoring of physiological data [25] [26] and can also be deployed to detect the behaviour of elderly people [27] [28]. With the help of

sensors, doctors may be able to identify predefined symptoms earlier [29] [30]. Also, due to remote monitoring via a wireless sensor network, the patients have an option to stay at home rather than at a treatment centre [31] [32]. Sensors may also be used for monitoring of patients and tracking of doctors inside a hospital. Sensors are also finding use in drug administration in hospitals for minimising the prescribing of wrong medicines to patients [33].

2.1.4 Home Applications

With the advancement of sensor technology, a number of applications have been proposed for home. It is proposed that sensors and actuators may be buried in home devices like refrigerator, and microwave oven etc. The sensors embedded in these devices can interact with each other and also with the outside world via the internet or satellite. The owner of the devices will then be able to operate these devices remotely [34]. Another scenario of creating a smart environment is described in [35]. This work describes a scenario of building a smart information environment center building in a campus at the beginning of a semester.

2.1.5 Other Commercial Applications

Some other commercial applications of sensors include environmental control of office buildings [36], interactive museums [36], detecting car thefts [37], inventory control, and vehicle tracking and detection [38].

In the following section, some examples of wireless sensor networks hosting multiple applications are discussed. These networks are referred as multi-purpose wireless sensor networks in literature [16].

2.2 Multi-purpose Wireless Sensor Networks

In the field of civil engineering, one of the most important applications of wireless sensor networks is assessment of the status of various types of bridges. A project called Sustainable Bridges has been implemented in Sweden to assess the railway bridges and to ensure that these bridges will be able to meet the present demands and also the future demands of traffic on the European Railway Network [39]. Sensors may be deployed to measure various characteristics like temperature, pressure, and vibrations. The sensors required to measure the various physical phenomena might be deployed at different locations. Some sensors may measure more than one phenomenon and may be common to more than one application.

Sharing of sensors by multiple applications is also described in [40]. This work describes an internet based data acquisition system in which various clients can issue requests to collect application specific data from sensors installed in remote locations.

A freight container monitoring scenario where a wireless sensor network is shared by multiple applications is described in [17]. Sensors can be deployed inside containers to serve multiple functions: to monitor environmental conditions in case the goods are perishable, and detect leakage in case of dangerous goods. Radio Frequency Identification (RFID) can also be used for inventory tracking by attaching sensors to the goods to be tracked [17]. Such a network can be used to track the containers during the journey, locate any lost containers, and also locate the location of a particular container in a stack. Each of these applications makes use of the wireless sensor network in a distinct way.

Another work on wireless sensor network being used as a shared resource is described in [41]. Due to scarcity of water in arid and semi-arid regions there is a lot of interest in projects focussing on reuse of treated wastewater. One such project has been deployed in Palmdale, California [41]. Irrigating with reclaimed water has lots of benefits; however it is important to test the quality of reclaimed water, as there could be chemical residues in the treated wastewater. The wireless sensor network installed at this site comprises soil moisture, temperature, and nitrate sensors. The network serves two important applications: first it ensures that all the environmental conditions are met with respect to the absence of harmful residues in the reclaimed water, and second it provides a feedback to a water control system to optimize water flow and minimize penetration of chemicals into the subsurface.

Sensor networks are also important in the monitoring of data centres to improve system reliability and reduce downtime. Downtime of a data center can prove very expensive, but can be reduced by continuous monitoring. The key areas of concern are: temperature, humidity, airflow, and power. Monitoring of temperature is important in order to avoid the operation of computer equipment at high temperatures over an extended period of time. It is important to monitor humidity to measure the moisture content in the air. Excessive moisture in air can cause condensation to build up on computer components leading to short circuits. As most devices draw cool air from the front and exhaust hot air at the back, it is necessary to ensure sufficient air flow in order to prevent warm air to be drawn to the next rack. Various types of sensors may be deployed at various different positions to measure these factors. SensiNet is one commercially developed solution for monitoring of data centres [42]. It supports multiple

applications on a single platform. Companies can install sensors to measure temperature, voltage, asset security, and contact closure. All these applications use the same network, gateway devices, and analytical software. Various components of the SensiNet system are shown in Figure 2-1 [42].

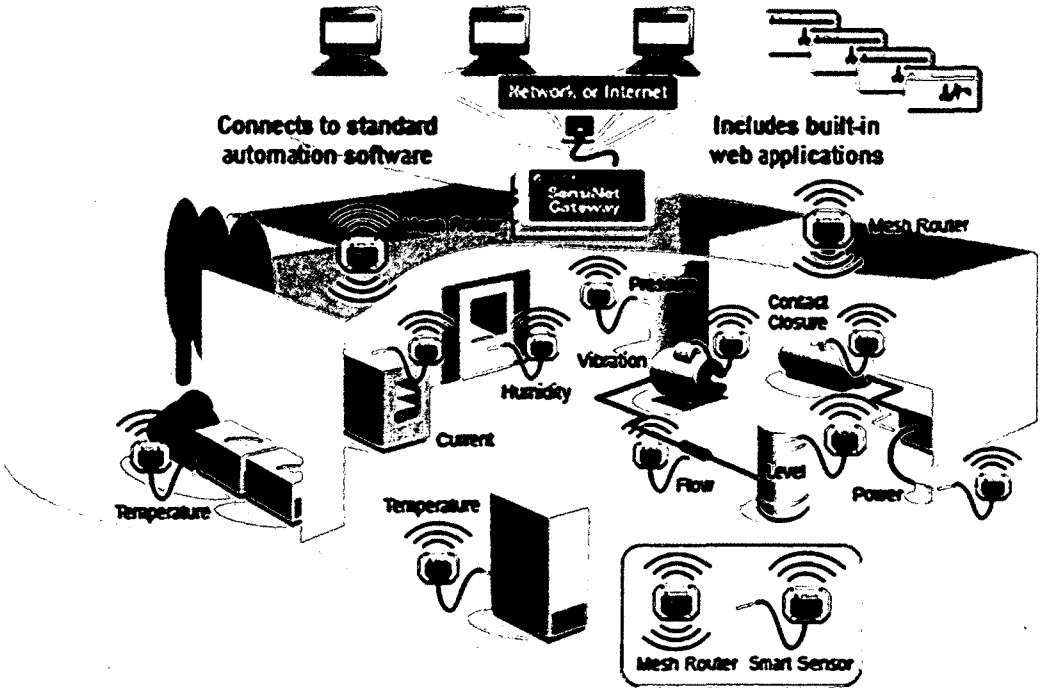


Figure 2-1: The SensiNet System (from [42])

In [43], the researchers propose a wireless sensor network supporting multiple applications to handle certain problems in a residential complex. The five applications address issues related to robbery, individual domiciles monitoring, and surveillance of patients, tracking of children and pets, and monitoring air pollution.

In [18], the authors discuss the importance of resource sharing for commercial deployment of wireless sensor networks as it reduces the cost of deployment and administrative costs. The authors propose a system that supports concurrent applications

on a single sensor node. As an example of concurrent applications supported by a wireless sensor network, the authors talk about a wireless sensor network in an enterprise building. Multiple departments and owners share the wireless sensor network. Each department runs its own application on the sensor network. This sensor network can support several applications concurrently: the facilities department monitors the temperature, luminance, and humidity by executing a monitoring application on a subset of uniformly distributed nodes. The building owner can deploy a structural health monitoring application on sensors located at critical parts of the building. A tracking application may be deployed to monitor the movement of personnel within the enterprise. There can be concurrent execution of these applications, both at the node level and the network level. At the node level, multiple applications may run on the same node concurrently. At the network level, different portions of the network (that may be overlapping) may serve different applications.

Another example of multiple applications sharing a wireless sensor network is discussed in [14]. There may be a sensor network in the battle zone to detect enemy vehicles. If friendly vehicles are equipped with RFID tags, the same network may be used to signal Friend or Foe in order to avoid casualties and attack on friendly vehicles.

2.3 Communication Architecture of WSNs: a High Level Perspective

Sensor nodes are scattered in the region of interest. The users submit an application request to the wireless sensor network via a task manager node. The application request is transmitted to the sink node via the internet or satellite. The scattered sensor nodes sense the desired data attribute and route the data to a sink node

via other sensors using a multi hop routing mechanism. The sink node may communicate with the task manager node via the internet or satellite. Such architecture is shown in Figure 2-2.

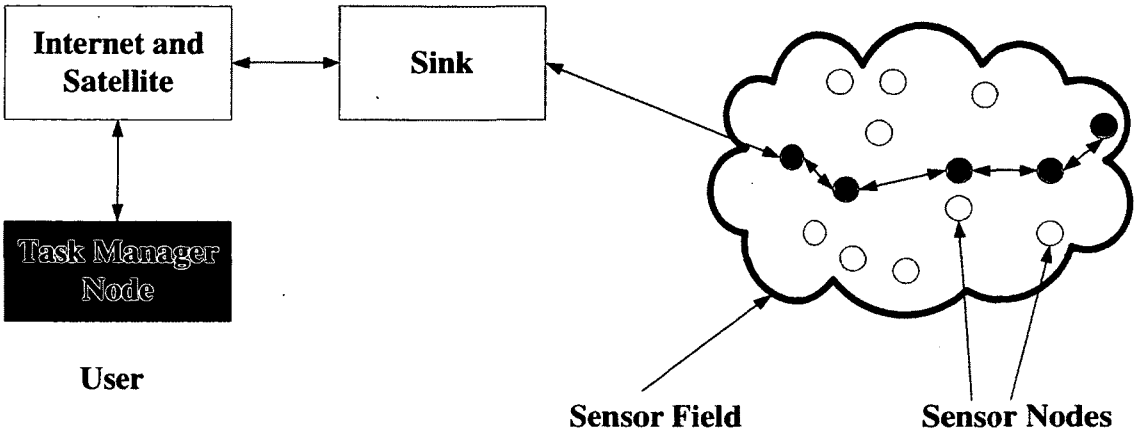


Figure 2-2: View of the Sensor Field (from [22])

The protocol stack used by the sensor nodes and the sink node for achieving this communication is shown in Figure 2-3. The protocol stack comprises of protocol layers and the management planes. The protocol layers and the management planes are explained briefly in the following sub-section.

2.3.1 Protocol Layers

The protocol layers are similar to the layered architecture in a wireless network: application layer, transport layer, network layer, data link layer, and the physical layer.

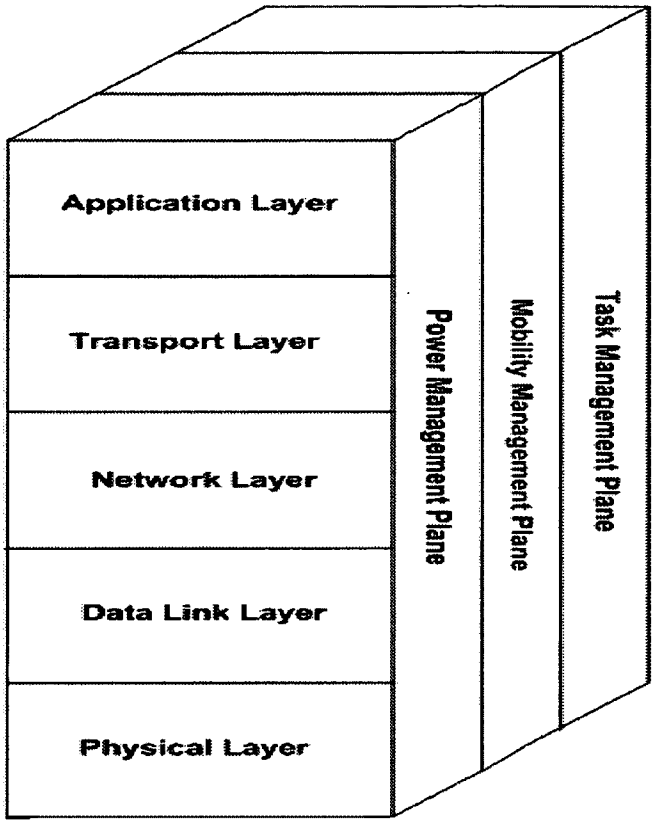


Figure 2-3: The Protocol Stack of Sensor Nodes (from [22])

2.3.1.1 Application Layer

Based on the sensing task, different kinds of application software can be used and built on the application layer. An application layer protocol makes the hardware and the software of the lower layers transparent to the application. Three possible application layer protocols have been proposed [22]: Sensor management protocol, Task assignment and data advertisement protocol, and Sensor query and data dissemination protocol.

Sensor networks comprise of sensor nodes that do not have global ids and will have to be addressed using a location-based or an attribute-based scheme. A sensor management protocol is required to address the following issues [22]:

- Introducing the rules related to a naming scheme based on attributes of sensors.

- Introducing the rules related to data aggregation in order to conserve sensor energy.
- Exchanging data related to the location finding algorithms.
- Time synchronization of the sensor nodes.
- Turning sensor nodes off whenever they are not being used.
- Introducing rules to query the sensor nodes.
- Adding security features to the sensor nodes participating in the wireless sensor network.

There are two ways by which the users can obtain the desired data from the sensors in a wireless sensor network. Either the users can issue queries to some or all of the nodes to get the desired data or the sensor nodes can advertise their data and the user nodes can query the data they are interested in. The task assignment and data advertisement protocol defines the methods by which the sensor data may be collected by user nodes. A sensor query and dissemination protocol provides an interface to users to issue queries based on attributes or the location of a sensor node. Sensor query and tasking language (SQTL) has been proposed to provide this functionality [44]. The research in this thesis is based on this model.

2.3.1.2 Transport Layer

It may not be possible to use standard transport layer protocols like TCP used in wired networks in wireless sensor networks [22]. Due to the limited power and memory resources of sensor nodes, it is difficult for sensor nodes to support an acknowledgement-based mechanism like TCP. For most of the applications involving wireless sensor

networks, the data collected from the sensor nodes is usually transported and used by wired networks. A split-level transport mechanism may be used with the TCP or UDP protocol implemented between the sink and the wired network and the communication between the nodes in the wireless network based purely on UDP [22].

2.3.1.3 Network Layer

As discussed earlier in Section 2.2, the sensors communicate with the sink via multi-hop communication architecture. The network layer takes into account the routing aspects in a sensor network. Adhoc routing techniques described in literature [45] may not meet the special requirements of a sensor network [22]. The design of routing algorithms in a sensor network should consider the following factors [22]:

- The routing algorithm should be efficient in terms of power consumption.
- Data can be aggregated before transmission if it does not affect the collaborative effort of the sensor nodes.
- Sensor networks are data centric in nature.
- Ideal sensor networks use an attribute based or location based naming scheme.

Some of the approaches that may be used to find an energy efficient route are [22]: maximum available power route (computed by taking the sum of available power of all the nodes along the route), minimum energy route (route that consumes minimum energy to transmit data from the source to the sink node), and minimum hop route (route with minimum number of hops from the source to the sink).

Some of the network layer schemes proposed for sensor networks are: Small Minimum Energy Communication Network (SMECN) [46], Gossiping [47], Sensor

Protocols for Information via Negotiation (SPIN) [48], Sequential Assignment Routing (SAR) [49], Low-energy Adaptive Clustering Hierarchy (LEACH) [50], Directed Diffusion [51], and Greedy Perimeter Stateless Routing (GPSR) [52].

The SMECN protocol [46] creates a sub graph that contains the minimum energy path. The number of edges in the sub graph is the same as the number of edges in the original graph and all the nodes that are connected in the original graph are also connected in the sub graph. In case of Gossiping [47], the data is sent to one randomly selected neighbour contrary to flooding where the data is sent to all the neighbours irrespective of whether it has received it before or not. In case of the SPIN protocol [48], data is transmitted to only those nodes that are interested in that data. In case of the SAR algorithm [49], data is routed from the sensor node to the sink based on the energy resources and additive QoS metric of each path, and the packet's priority level. The additive QoS of each path is calculated by adding the QoS of the nodes along the path. Sensor nodes that have low throughput and high delay have a low QoS. The LEACH protocol [50], forms clusters of sensor nodes in order to minimise dissipation of energy. In case of Directed Diffusion [51], the data is sent from a sensor node to the sink along the gradient path. The gradient path is established during interest dissemination from the sink to the sensor nodes. Greedy Perimeter Stateless Routing (GPSR) uses greedy forwarding to send packets to nodes that are always progressively closer to the destination [52]. GPSR makes greedy forwarding decisions using only information about a node's immediate neighbors in the network topology. Hence, GPSR allows building of networks that cannot scale by simply using existing routing algorithms for wired or wireless networks. Researchers have demonstrated the suitability of the GPSR protocol

for wireless sensor networks [52]. The research in this thesis assumes GPSR as the underlying routing protocol.

2.3.1.4 Data Link Layer

The main function of the data link layer is multiplexing of data streams, detection of data frames, medium access control (MAC), and error control [22]. MAC layer protocols broadly fall into two categories: Contention based and Schedule based.

Contention Based Versus Schedule Based MAC Protocols: The schedule based protocols, such as Time Division Multiple Access (TDMA), have an advantage over the contention based protocols with respect to energy conservation. This can be attributed to the fact that the fraction of the time the radio is transmitting is reduced and there is no contention based overhead and collisions. However, the schedule based protocols do not adapt easily to any changes in the number of nodes in a WSN. The scalability of a schedule based MAC protocol is not as good as that of a contention based MAC protocol as new nodes may be added or old nodes may be removed from a WSN [53]. Also, contention based protocols may be more suitable for networks with less traffic due to lesser collisions as compared to networks with heavy traffic. Wireless sensor networks usually fall into this category. Also, in our research we aim to provide best effort service to the users of applications aiming to reduce the overall mean response time to the users of applications. Such a service is easily applied to applications that are not mission critical and delay tolerant. Examples of such delay tolerant scenarios are environmental monitoring applications where the application can still function as desired even if some data requires a long time for delivery. For such delay tolerant applications, the data delivery

requirements can be relaxed in both the time and reliability domains. A contention based protocol is a suitable choice for such a scenario. A typical example in the category of contention based MAC protocol is the IEEE 802.11 protocol [54].

Contention Based MAC Protocols for Wireless Sensor Networks: The IEEE 802.11 protocol is widely used in the area of wireless adhoc networks because of its simplicity and robustness to the hidden terminal problem. In wireless networks, the hidden terminal problem occurs when a node is visible from an access point, but not from other nodes communicating with the said access point. Various researchers have talked about unsuitability of the 802.11 in the area of wireless sensor networks due to its high power consumption and have worked in the field of contention based MAC protocols suitable for a WSN. SMAC [55] is one of the pioneer MAC protocols in the category of contention based MAC protocols for a WSN. In the SMAC protocol, the listen time of the nodes is reduced by letting the node go into periodic sleep mode. The duration of listening and sleep can be selected according to the application scenario. If multiple neighbours want to talk to a node, they need to contend for the medium when the node is listening. The contention mechanism is the same as that in IEEE 802.11 and uses Request To Send (RTS) and Clear To Send (CTS) packets. The node that first sends out the RTS packet wins the medium, and the receiver will reply with a CTS packet. After they start data transmission, they do not follow their sleep schedules until they finish transmission. With this protocol, latency is increased due to the periodic sleep of each node. Moreover, the delay can accumulate on each hop. So the latency requirement of the application places a fundamental limit on the sleep time. The SMAC protocol tries to avoid

overhearing by letting interfering nodes go to sleep after they hear an RTS or CTS packet. An extra delay in SMAC is caused by a node's periodic sleeping. When a sender gets a packet to transmit, it must wait until the receiver wakes up. This is called sleep delay since it is caused by the sleeping of the receiver. It has been shown in [55] that the energy consumption of the SMAC protocol is less than the energy consumption of the IEEE 802.11 protocol. However, the latency produced due to the SMAC protocol is higher than the latency caused by the 802.11 protocol. Various other contention based protocols have been proposed. These protocols try to reduce the forwarding delay by introducing an adaptive sleep period. Some examples are: SMAC-AL [56], T-MAC [57] and DSMAC [58]. A complete survey of MAC protocols is presented in [53]. The survey discusses the various contention based and schedule based protocols and the type of applications they would support with delay and reliability guarantees. The use of IEEE 802.11 protocol in WSNs is described next.

IEEE 802.11 MAC Protocol for Wireless Sensor Networks: As mentioned earlier, various proprietary solutions exist for wireless sensor networks but there is no standards based solution. Various MAC layer protocols have been proposed for wireless sensor networks. There is also a lack of standardization at the physical layer and the physical sensor hardware. At the same time IEEE 802.11 networks are widely deployed for commercial, public, and consumer applications, having met the criteria for such requirements as security, manageability, and cost. The use of IEEE 802.11 has been limited in wireless sensor networks due to the high power consumption and because many applications require years-long battery life for sensor node devices. New 802.11 designs that meet

battery life requirements for sensor nodes enable organizations to capitalize on the their existing 802.11 infrastructure investment with the benefits that generally come with standards-based solutions, such as lower costs due to economies of scale, interoperability, ease of use, and availability of mature management tools, as well as the ability to support other 802.11 client devices [59]. As the 802.11 networks are already existing, 802.11 sensor nodes can be easily added without disturbing the existing distribution infrastructure. There can be numerous examples of sensor networks that could make use of an existing 802.11 network. SeaWorld in Orlando, FL, for example, utilizes 802.11 to connect a small number of previously isolated buildings located on "islands" to its centralized energy management system. A sensor network in an enterprise building could be based on the IEEE 802.11 standard. Home security, comfort, and entertainment systems using 802.11 may be installed faster and cheaper using home 802.11 networks rather than a dedicated network, and can be made remotely accessible over the internet without special gateway interfaces to proprietary systems. These are only a few examples of applications where an existing 802.11 network may be used.

GainSpan has recently introduced a chip that contains an 802.11 radio and also has low power consumption suitable for wireless sensor networks [60]. The solution offered by GainSpan can be incorporated wherever an 802.11 network is available or wherever an 802.11 network can be easily deployed. A number of examples where WSNs can use the existing 802.11 networks or where 802.11 networks can be easily deployed are presented.

- It can be incorporated into products that may be used in industrial motor monitoring.

- It can be used for monitoring various phenomenons in buildings and data centres.
- It can be used in food and drug manufacturing industry to monitor temperature of goods through their supply chain.
- It can be used in auto-manufacturing plants to track vehicles during production,
- It can be used in hospitals to track patients, wheelchairs, diagnostic equipment, and staff.
- Street lights and traffic lights may be monitored in public metro areas.

All these applications may be able to use an existing 802.11 infrastructure. There may be scenarios where other proprietary solutions may be required to support an application's requirements, but there is a vast variety of applications that will be supported by the 802.11 networks.

We have assumed the 802.11 MAC protocol in our research. A few experiments have also been performed by simulating a protocol similar to the SMAC protocol [55]. The results for these experiments are presented in Chapter 4. The results demonstrate that the relative performance of the proposed algorithms is not affected by the selection of the MAC protocol.

2.3.1.5 Physical Layer

The physical layer is responsible for selection of frequency of signal, generation of carrier frequency, signal detection, modulation and data encryption [22].

2.3.2 Management Planes

The management planes: power management plane, mobility management plane, and the task management plane are orthogonal to the protocol layers and can be implemented in any of the layers [22].

2.3.2.1 Power Management Plane

Power Management Plane deals with the ways to manage the power of a sensor node. For example, a sensor node may switch off its receiver after receiving a message from one of its neighbours in order to avoid receiving duplicate messages [21]. Also, when the power of a sensor node falls below a threshold, the sensor node may broadcast to its neighbours that it will no longer participate in routing activities, and the remaining power will be used only for sensing activities [21].

2.3.2.2 Mobility Management Plane

The mobility management plane deals with issues relating to the mobility of sensor nodes. In case of applications in which the sensor nodes are mobile, it is important that up to date information about the position of sensor nodes is maintained so that any sensor node knows at any time about its neighbouring nodes and is able to find a path to the sink node.

2.3.2.3 Task Management Plane

All sensors may not be active at the same time in a particular region. While a particular node performs the sensing task, other nodes in the same region may be in sleep mode. A node may be selected to perform a sensing task based on its power level with respect to the other nodes [21].

2.4 Design Issues and Challenges in a Wireless Sensor Network

A wireless sensor network comprises sensor nodes that have limited resources compared to nodes in a wired grid. Sensor nodes have limited processing power, battery resources, and storage capacity. Also, the sensor nodes communicate via unreliable and low bandwidth wireless links. These limitations of sensor nodes give rise to various design issues and challenges in a wireless sensor network [61]. Some of the design issues and challenges in a wireless sensor network are discussed [61].

2.4.1 Grid APIs for sensors

Grid APIs and standards are too complex to be loaded on sensor nodes due to their limited resources. Architecture described in [61] uses a proxy as an interface between the sensor nodes and the computing grid.

2.4.2 Network Connectivity and Protocols

The devices participating in a wired grid are connected via fast and reliable wired connections. Contrary to this, sensor nodes participating in a sensor network communicate via low bandwidth and unreliable wireless connections. Also in applications involving mobile nodes, connections may be dynamic in nature. Thus the sensor grid should be able to adapt to unexpected network disconnections.

2.4.3 Scalability

It should be possible to add another sensor network to an existing wireless sensor network with the aim of serving another application without much change in the software architecture. This enables a wireless sensor network to comprise multiple virtual sensor networks with each virtual sensor network serving a particular application.

2.4.4 Power Management

Sensor nodes have limited power resources as compared to their counterparts in a wired grid. Resource management algorithms in a sensor grid should be based on methods to conserve battery power of sensor nodes in order to enhance the network life span of a sensor grid.

2.4.5 Scheduling

Scheduling of sensor nodes can be viewed from two perspectives. Some researchers refer to scheduling in the context of alternating sleep and idle cycles of sensor nodes in order to conserve power. For example, in an application-supporting target tracking, sensor nodes that are located away from the target may be selectively switched off. Also, all nodes in a particular geographic area may not be required to be active at the same time.

Sensor grids are data-centric in nature. The sensors collect data and via a multi-hop communication mechanism transmit the collected data to a sink node. Also, as discussed in Chapter 1, a wireless sensor network may comprise of various virtual sensor networks, each corresponding to a specific application. The sensor nodes may be a part of more than one virtual sensor network if they serve more than one application. For a wireless sensor network serving more than one application, scheduling refers to the order in which the requests corresponding to applications may be executed on the sensor nodes. The aim of any scheduling algorithm in a sensor network is to improve the overall mean response time to the users of the applications. A scheduler in a sensor grid will have to take into account the characteristics of applications while making a scheduling decision.

Sensor jobs differ from computing jobs in various aspects. Sensor jobs are not multitasking in nature and cannot be pre-empted. Once started, a sensor job runs to completion. Also, the duration of sensor jobs has to be specified in advance and some sensor jobs may also require specific time slots for execution.

2.4.6 Security

Wireless sensor networks have a higher security threat as compared to wired grids mainly due to the unreliability of wireless communication [61]. Some of the common security problems are compromising and tampering of sensor nodes, eavesdropping on sensor data, and denial of service attacks. These problems need to be addressed and some of the techniques that may be used to address these problems are authentication of sensor nodes, encryption of sensor data, adding security at the MAC and routing layers.

2.4.7 Availability

Due to the limited resources of the sensor nodes and unpredictability of wireless communication, applications running on sensor nodes have a high probability of failure. It should be possible to transfer a job from a failing sensor node to another node. Methods should be in place to enable recovery of the failed jobs.

2.4.8 Quality of Service

It is difficult to ensure quality of service to applications running on a wireless sensor network due to the unpredictable nature of wireless communication and limited resources of sensor nodes.

Wireless sensor networks and grid computing are two promising technologies and when integrated into a sensor grid, the potential of both the technologies: wireless sensor

network and computing grid is greatly enhanced. The architecture of a sensor grid must address the issues and challenges discussed in this section. Some relevant architectures are described in Chapter 3.

Based on these limitations of sensors, the sensor resources must be managed efficiently in order to improve the overall performance of the wireless sensor network. Some of the work done in the field of resource management in sensor networks is described next.

2.5 Resource Management in Wireless Sensor Networks

This section presents the state of the art in resource management in wireless sensor networks. The challenges of supporting multiple applications in a wireless sensor network are presented.

Most of the work done in the field of wireless sensor networks aims at ways to reduce the energy consumption of the sensors comprising the wireless sensor networks and also provide reasonable turnaround times to the users of the applications supported by the wireless sensor network. In case of wireless sensor networks communication time plays a more significant role as compared to the computation time. Some researchers have worked on methods to reduce the communication time by aggregating data at sensor nodes in order to schedule lesser number of subsequent transmissions [62]. The authors demonstrate that significant amount of energy savings can be attained by aggregating data in high density wireless sensor networks without affecting the latency adversely. In order to conserve energy, some researchers have worked on strategies to put sensors to sleep when they are not needed. Researchers suggest that the life of a network can be

increased by putting redundant sensors (i.e., sensors not needed to provide coverage of all targets) to sleep and awaken these sleeping sensors when they are needed to restore target coverage [63] [64]. Sleeping sensors are inactive while sensors that are awake are active. Inactive sensors consume far less energy than active ones. In [65], the authors propose the concept of virtual clusters, a mobile data collector, and also dynamically scheduling the sleep and wake up cycles of sensors. Instead of relaying the sensed data to a sink node, the nodes of a cluster relay the data to the cluster head, which in turn transmits the collected data to a mobile data collector. Nodes within a cluster communicate via a multi hop communication mechanism. The selection of cluster head is also dynamic and is based on the position of a node relative to the mobile data collector. The node that is closest to the mobile data collector is selected as the cluster head. While the mobile data collector is collecting data from any given cluster, the nodes in clusters located farther away from the mobile data collector may be put in the sleep mode to conserve energy. Some researchers propose partitioning the set of available sensors into disjoint sets such that each set covers all targets. Several decentralized localized protocols to control the sleep/awake state of sensors so as to increase network lifetime have been proposed [66].

A wireless sensor network may be dedicated to a single application or may comprise of multiple virtual sensor networks with each virtual sensor network serving a given application at a given time. In most of the work described in literature, researchers focus on wireless sensor networks being dedicated to a single application. However, it is suggested that no single application would warrant widespread deployment of a wireless sensor network [67]. The deployment of WSNs hosting multiple applications provides more effective utilization of the WSNs [67]. With the advancement of sensor technology

and with the aim to increase the cost efficiency of wireless sensor networks, multi-purpose wireless sensor networks serving multiple applications are replacing sensor networks dedicated to a single application [67] [16] [14]. When a wireless sensor network is shared by more than one application, management of sensor resources becomes a challenging task. Resource management of a WSN comprises of two major functions: *Scheduling* and *Allocation*. Scheduling determines the order in which the requests belonging to various applications would be executed. Allocation refers to the task of selection of sensor nodes from a set of available nodes that will be used for executing the application request.

2.5.1 Scheduling in WSNs Hosting Multiple Applications

Little work has been done in the field of scheduling multiple applications over a wireless sensor network. In [68], the researchers deal with jobs from different applications and different deadline constraints. Such operating scenarios often arise in wireless video surveillance and target detection applications running on sensor networks [69]. The jobs can run on any node and only the total number of sensors on which the jobs of a particular application are to be run is specified. The algorithms have been adapted to the characteristics of sensor jobs i.e. sensor jobs are not pre-emptible. The algorithms tested are First Come First Served (FCFS), and the non-pre-emptive versions of Earliest Deadline First (EDF), Least Laxity First (LLF), and Shortest Job Next (SJN). To the best of our knowledge no other scheduling algorithm exists in literature that focuses on providing best effort service on WSNs hosting multiple applications that this research focuses on. Also, none of the work discussed in literature uses knowledge about

the characteristics of an application or network in order to make a scheduling decision. Different applications may have different attributes in terms of the number of sensors required, data size of request and response messages, and location of the sensors. Some scheduling algorithms take scheduling decisions without any knowledge of the system or the application. A typical example of a knowledge free scheduling algorithm is the First Come First Served scheduling algorithm. In case of the First Come First Served scheduling algorithm, the jobs are scheduled based on the order in which they arrive. Another example of a knowledge free scheduling algorithm is the round robin scheduling algorithm, in which all jobs get an equal share of the resource. However, as the sensor jobs are non-pre-emptible in nature, the round robin scheduling technique will not work in a wireless sensor network.

Knowledge free algorithms have fewer overheads, however knowledge based algorithms may give better performance than the knowledge free algorithms. The knowledge based scheduling algorithms may use the knowledge of the system and/or the application in order to make a scheduling decision. For scheduling requests from multiple applications in a wireless sensor network, scheduling algorithms based on the knowledge of the application and the network are proposed in this research. These algorithms are described in detail in Chapter 3.

2.5.2 Allocation in WSNs Hosting Multiple Applications

The problem of resource allocation has been addressed by various researchers in various kinds of networks like wireless networks [70], grid [71], and real time computing systems [72]. Resource allocation in wireless sensor networks has also attracted

significant attention from researchers. As many sensing applications are designed to estimate spatially correlated phenomenon, subsets of sensor nodes could be allocated to multiple contending applications. Wireless sensor networks comprise of resource constrained sensors that have limited resources in terms of energy, memory, and bandwidth. Considering the resource constrained nature of the sensor nodes in the WSN, the main objective of resource allocation in wireless sensor networks is to improve the lifetime of the wireless network. In [73], the researchers propose an online allocation algorithm to dynamically reconfigure the network whenever a hot spot occurs (detection of an intruder, for example) or the state (sleep or awake) of a sensor node changes [73]. In another work [74], the authors propose a strategy for selection of a set of sensors based upon a trade-off between application-perceived benefit and energy consumption of the set of sensors that will be used to serve an application. In [75], the authors study the problem of resource allocation in a WSN in the context of mobile target tracking. Upon detection of an event, a set of sensors which can contribute towards tracking of the target forms a cluster while all other sensors are put in sleep mode. The membership of the cluster is updated based on the movement of the target. Energy harvesting techniques have opened up a new domain in the field of resource allocation in WSNs [76]. The work done in this area is based on the principle that rate of energy consumption of sensors should not exceed the energy harvesting rate in order to prevent the depletion of energy of sensors. In [76], the authors propose an algorithm which adapts the sampling rate of sensors with the objective of maintaining the battery energy at a target level. Researchers have also worked on allocation algorithms that are based on a trade-off between the energy

consumption of sensors and the quality of the monitored information that may be dependent on the period of measurement updates [77].

However, most of these research works have focused their attention on wireless sensor networks serving a single application. Very few existing works have addressed resource allocation in shared wireless sensor networks. In [78], the authors propose a Utility-based Multi-application Allocation and Deployment Environment (UMADE), which is an integrated system for application deployment in a shared sensor network (UMADE). In [78], the authors have worked on allocation algorithms to maximize the total system utility which is the sum of weighted utility of the applications to be deployed given the memory constraints of the sensors. The profile of the application contains the application code, its weight, memory requirements, and the utility function. The utility function of an application defines the acceptable Quality of Monitoring (QoM) values for an application and the corresponding utility values [72], [79], [80]. The UMADE system automatically generates QoM values with respect to the subset of sensor nodes on which the application may be deployed. A greedy allocation algorithm is invoked whenever a new application needs to be deployed. The algorithm uses the information provided by the application profile, and application QoM values with respect to the subset of nodes and attempts to maximize the total system utility. The authors demonstrate an improvement in system utility obtained with the proposed greedy allocation algorithm when compared to a random allocation algorithm [78]. In another work [81], the researchers aim to enhance the total weighted quality of monitoring of the deployed applications subject to the memory and bandwidth constraints on the sensor nodes. The performance of the proposed algorithm is compared to a bin packing algorithm [81]. This

work is complimentary to [78]. In [68], the authors have discussed a few load balancing techniques in the context of applications with firm deadlines. The research in this thesis aims to provide a best effort service to the users of the multiple applications deployed in the WSN while trying to increase the lifetime of the wireless sensor network. The proposed allocation algorithms enhance the lifetime of the wireless sensor network while trying to balance the load amongst the sensors. These algorithms take allocation decisions based on the attributes of applications and the network and are described in detail in Chapter 3.

Chapter 3: Simulation Model & Resource Management Algorithms

This chapter describes the system architecture, experimental set-up, the simulation model and the simulation toolkit used. It presents the various allocation and scheduling algorithms proposed in this research in detail. It also provides an overview of the system parameters, workload parameters, and performance measures.

3.1 System Architecture

Software tools are essential for the management of wireless sensor networks. Some of the sensor management software that exist are MoteLab [82], EmStar [83], and Kansei [84]. These softwares enable a user to program the sensors, create jobs for sensors, collect data from the sensors, and perform simple administrative functions. However these management softwares manage a standalone wireless sensor network and not a sensor grid in which the wireless sensor network is required to be interfaced with a computing grid. In this research, the focus is on resource management of sensor nodes comprising a WSN that is interfaced with a computing grid. The users of applications in a computing grid issue requests for an application that is hosted on the WSN.

Due to the growing popularity of sensor grids, researchers have worked on architectures that will enable wireless sensor networks to interface with the computing grid. Some of the architectures described in literature are application specific while some of the architectures are too complex to be loaded on simple sensors [85].

The system architecture considered in this research is based on Scalable Proxy-based Architecture for Sensor Grid (SPRING) proposed in [61]. It is independent of the

applications to be supported by the wireless sensor network, and can also be implemented with the simplest of sensor nodes with limited capabilities. As shown in Figure 3-1, in the SPRING framework, a proxy system is used as an interface between the computing grid and the wireless sensor network. The SPRING system is based on a layered architecture approach. The layers represent the software components that are used to build a sensor grid. The layers at the user node, proxy, and at the computing resource as shown in Figure 3-1 are discussed next [61].

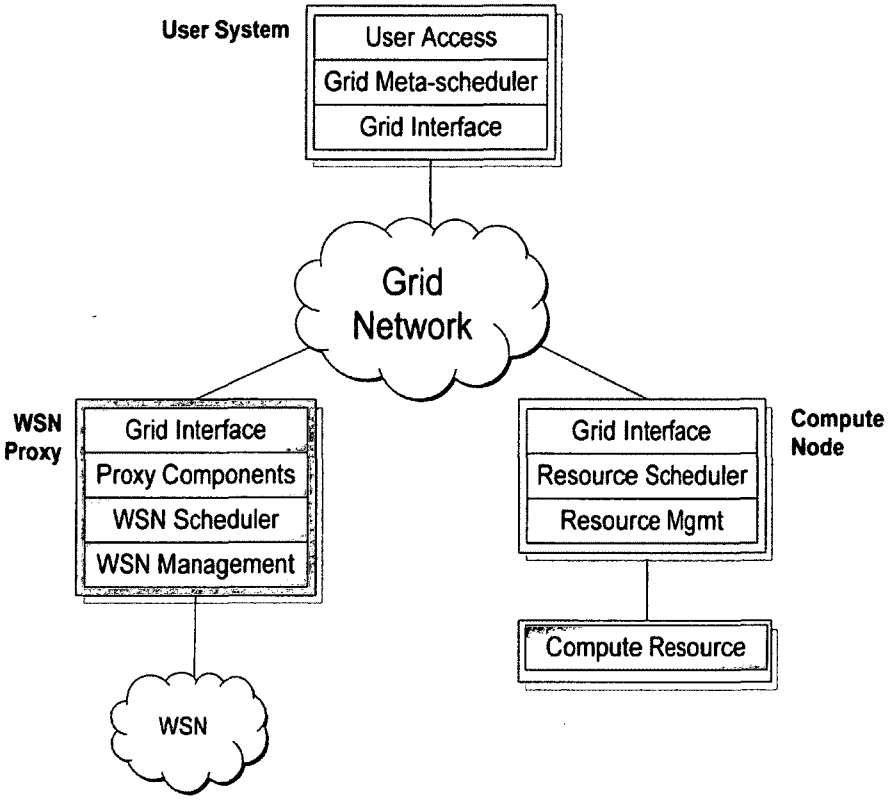


Figure 3-1: The SPRING Framework (from [61])

3.1.1 User System/Node

The User System comprises of a User Access layer, a Grid Meta-scheduler, and a Grid Interface. The User Access layer at the user node provides an interface as a grid

portal or a workflow management tool that enables the users to issue application requests to the sensor grid. An application request may comprise of sensor job requests required to collect data from the wireless sensor network or computational job requests required to process sensor data. A Grid Meta-Scheduler schedules and routes application requests according to the required resources. The Grid Interface layer provides middleware like the Globus toolkit [86] to interface with the grid.

3.1.2 Proxy

The proxy exposes the functionality of sensor nodes to the rest of the grid. Despite their resource constraints, the sensor nodes may be allocated as any other resource available on the grid. As shown in Figure 3-2, the proxy comprises of a WSN Management layer, a WSN scheduler, a Grid Interface layer, a Data Management component, an Information Services component, a WSN Connectivity component, a Power Management component, a Security component, an Availability component, and a Quality of Service Component.

The WSN Management layer at the proxy provides protocols and Application Programming Interface (API) to access and manage the sensors in the underlying wireless sensor network.

The WSN scheduler is responsible for scheduling the requests for applications on the wireless sensor network.

The Grid Interface layer supports standard middleware required to interface the wireless sensor network with the rest of the grid.

The Data Management component translates the data obtained from sensors to a standard Open Grid Services Architecture (OGSA) [87] format such as (Extensible Markup Language (XML) or any other format desired by the user.

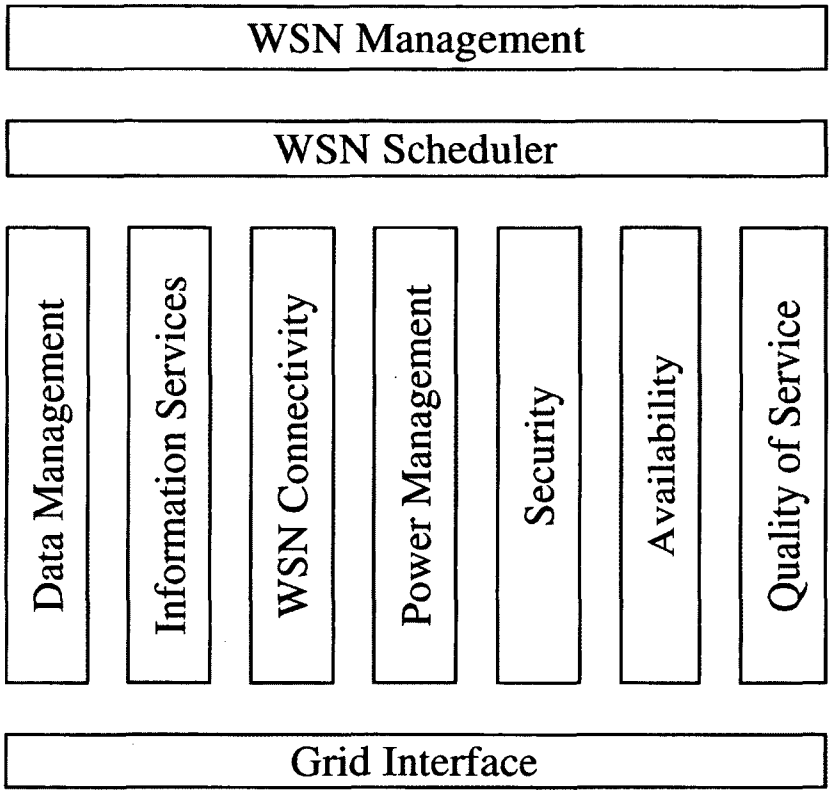


Figure 3-2: The Proxy Software Architecture (from [61])

The Information Services component manages the discovery and monitoring of the sensor resources in the WSN. Static and dynamic information on sensor resources can be queried and used by the WSN Scheduler.

The WSN Connectivity component provides services to maintain connectivity between the sensor nodes comprising the WSN. It maintains routing information of sensor nodes in a cache, buffers the transmission of sensor data, and also manages the ad hoc sensor links in a WSN.

The Power Management component keeps track of the power consumption of the sensor nodes. The WSN Scheduler works with the Power Management Component in order to conserve power of the sensor nodes.

The Security component implements OGSA-compliant grid security technologies to perform authentication between the proxy and the sensor nodes.

The Availability component provides services to monitor the sensor nodes that are available for execution of requests in a WSN. It monitors the sensor nodes so that the jobs may be migrated from sensor nodes with failing hardware or lower power level to sensor nodes with higher power levels. It works closely with the WSN Scheduler.

The Quality of Service (QoS) component supports the provisioning of QoS in the sensor grid. Based on the QoS requirements of an application, the QoS component performs the reservation and allocation of sensor resources.

3.1.3 Compute Node

A Compute Node comprises of a Resource Management layer, a Resource Scheduler, and a Grid Interface layer. The Resource Management Layer provides an API to manage and access the resource. The Resource Scheduler implements a local scheduling algorithm at the resource. Similar to the Grid Interface layer in other components, the Grid Interface layer provides middleware to interface with the grid.

J-Sim, the simulation toolkit used in building the simulation model is described next.

3.2 J-Sim: A Tool for Simulating Wireless Sensor Networks

The simulator used in this research is developed using J-Sim [88]. The J-Sim project has been supported by National Science Foundation (NSF). J-Sim [88] enables application development based on autonomous component architecture. The components are written in Java and a scripting language is used to glue the components together and define the operation of the system. J-Sim supports a few scripting languages: Tcl, and Tcl/Java (Jacl) that enables manipulation of Java objects in a Tcl environment.

3.2.1 The Wireless Sensor Network Package in J-Sim

J-Sim [88] has a separate package to simulate a wireless sensor network. The package for simulating wireless sensor networks in J-Sim is based on a sensor function model and power model described in [89] [90]. The sensor function model represents a software abstraction for all the sensor functionality modules. It includes the network protocol stack, user applications, sensor protocol stack and the middleware. The power model simulates the power consumption of hardware like the CPU, and the radio. The power model also simulates the battery of a sensor as an energy source [89]. Three main types of nodes can be simulated by the wireless sensor network package of J-Sim: sink nodes, sensor nodes, and target nodes. The target nodes transmit the stimulus (signal) to the sensor nodes, which finally transmit the signal to the sink node via other sensor nodes using a multi-hop communication mechanism. The sink node receives the data sent by all the sensor nodes. In this research, the proxy represents a sink node. A sensor node comprises two protocol stacks: a sensor protocol stack for communication between the target nodes and the sensor nodes, and a wireless protocol stack for communication

between the sensor nodes and the sink. The sensor function model consists of the sensor protocol stack and the wireless protocol stack. J-Sim provides full functionality to implement the sensor function model and the power model. The sensor function model and the power model are shown in Figure 3-3.

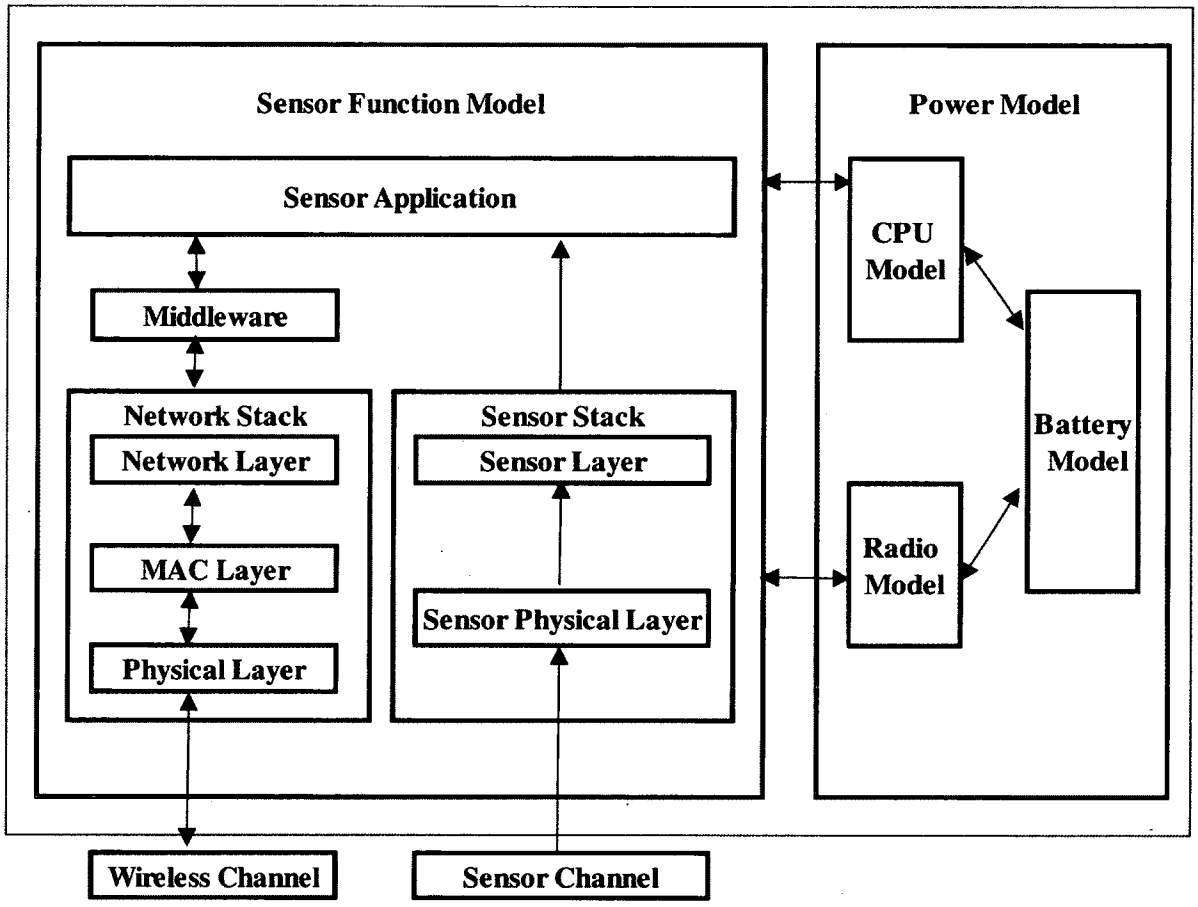


Figure 3-3: Sensor Function Model & Power Model of a Sensor Node (from [88])

The power model consists of a battery model, a CPU model, and a radio model. The battery model represents the energy producing components in a sensor node, battery for example. The CPU model and the radio model represent the energy consuming components of a sensor node. Sensor Channel is the medium through which the sensor nodes detect the events or the phenomenon. The layers in the sensor stack are responsible

for detecting and processing the signal coming from the sensor channel. The wireless channel represents the medium through which the sensor nodes transmit the signal to the sink node using a multi-hop wireless communication mechanism. The Network Stack, as shown in Figure 3-3, comprises the physical layer modelling a wireless card, the MAC layer implementing the MAC 802.11 protocol, and the network layer implementing the routing mechanism. The sensor application layer and the middleware co-ordinate the sensor stack and the network stack.

3.3 Simulation Model

In the simulation model presented in the thesis, target nodes (described in Section 3.2.1) have not been considered as the communication between the target nodes and the sensor nodes will not affect the performance of the scheduling algorithms that are analysed in this research. As mentioned in Section 3.1, the user nodes interact with the wireless sensor network via a proxy-based architecture. The simulation model is shown in Figure 3-4. The entities that have been implemented in the simulator are described next.

3.3.1 Traffic Source

Sensors may be programmed to collect, queue, and transmit sensory data at fixed frequencies that match the requirements of the applications. However, application users may also issue queries to get information from the WSN. The research in this thesis is for systems in which users issue queries to get information from the WSN. Sensor Query and Data Dissemination Protocol (SQDDP) provides user applications with interfaces to issue and respond to queries and collect incoming replies [91]. Sensor query and tasking language is used by users of applications to issue queries to collect data from sensors

[91]. In this research, the user requests submitted to a computing grid are simulated by a traffic source. The traffic source generates the traffic from users that submit requests that correspond to the applications supported by the underlying wireless sensor network. The stream of requests arriving at the proxy (see Figure 3-4) follows a Poisson process. The terms query and request are used interchangeably in this document.

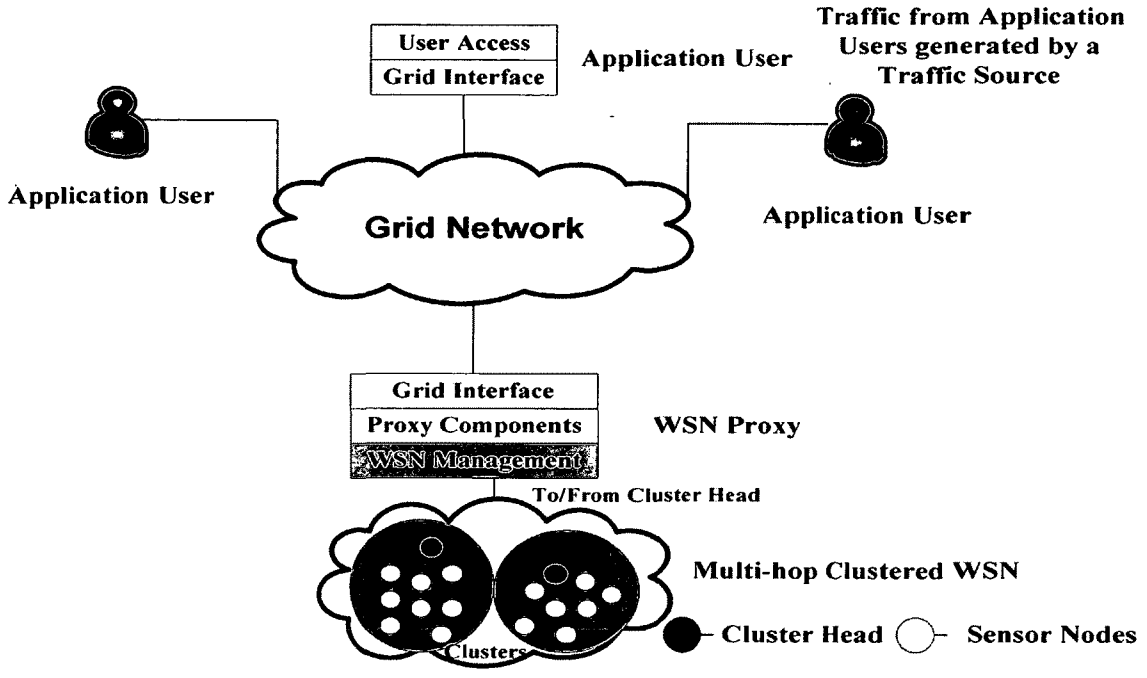


Figure 3-4: The Simulation Model

3.3.2 Proxy

The proxy acts as an interface between the computing grid and the wireless sensor network. The algorithms proposed in this research are deployed at the proxy. This can be attributed to the fact that sensors have limited energy and it is important to conserve their energy in order to increase the lifetime of the wireless sensor network.

3.3.3 Cluster Head

We assume that the wireless sensor network is organized in clusters, with each cluster having a cluster head. The cluster heads receive requests from the users via a proxy. The proxy creates a separate request message for each sensor on which the application needs to be run and transmits it to the cluster head. The cluster head then transmits the request message to the relevant sensor. After processing the request, the sensor sends the reply back to the cluster head, which in turn sends the reply back to the proxy. A multi-hop cluster is assumed in this research. Researchers have worked on various algorithms for formation of multi-hop clusters [92]. The formation of clusters is beyond the scope of this research.

3.3.4 Sensors

The wireless sensor network comprises of sensors. Sensor nodes of a particular cluster transmit the sensed data to the respective cluster head via other sensors using a multi-hop communication mechanism.

3.3.5 Wireless Channel

Communication amongst the sensors comprising the WSN is via a wireless communication channel following the IEEE 802.11 Media Access Control (MAC) protocol. As the 802.11 protocol can make use of the existing wireless infrastructure, various researchers have recommended the 802.11 protocol in the context of WSNs [59]. As described earlier in Chapter 2, to account for the resource constrained sensor nodes, chip sets supporting the 802.11 protocol and with low power consumption have been introduced [60]. Also, a large number of wireless sensor networks including wireless

sensor networks used in buildings, data centres, hospitals, and auto manufacturing plants are based on the 802.11 standard.

The 802.11 protocol is a contention based protocol that is suitable for our research that aims to provide best effort service to the users of applications as compared to schedule based protocols that are more suitable for mission-critical applications that have hard deadlines [53]. Various researchers have proposed contention-based MAC protocols based on the 802.11 protocol, but have lower power consumption as compared to the 802.11 protocol. SMAC [55] is one of the pioneer MAC protocols in the category of contention based MAC protocols for a WSN. The SMAC protocol has functionality similar to the 802.11 protocol with the sensor nodes having a periodic sleep mode in order to conserve the energy of the resource constrained sensors. The sleep mode adds more latency to the messages being transmitted in the WSN. In order to verify the performance of the proposed algorithms with a protocol similar to the SMAC protocol, an abstraction of the SMAC protocol is also created in our simulation model. The overall mean response time of the applications increases for all the scheduling algorithms with the SMAC protocol (discussed in Section 2.3.1.4). However, the relative performance of the proposed algorithms is the same as with the 802.11 protocol.

In this research, a WSN comprising a single cluster with 100 sensor nodes that are uniformly distributed in the geographic area of interest is simulated. Scenarios with both planned deployment of sensors and random deployment of sensors are considered in this research. The sensor nodes are spread up to a maximum distance of 4 hops from the cluster head. The sensor nodes that are not directly connected to the cluster head relay the data to the cluster head using other sensors and a multi-hop communication. A similar

architecture has been used by other researchers [92]. Sensor transmissions may affect nearby sensors. In a shared wireless sensor network, sensors close to a transmitting sensor will not be able to transmit as long as the shared medium is sensed busy.

3.3.5.1 Communication between the Proxy and the Wireless Sensor Network

Deployment of sensors in a wireless sensor network may fall into two categories: Random or Planned. In planned deployment the location of the nodes can be carefully selected. There are various environments such as buildings and known fields where a planned deployment is targeted [93]. Various researchers are working on algorithms for effective planned deployment of sensor nodes in order to provide full coverage and full connectivity [94]. For various environment monitoring applications, a planned deployment may not be possible as sensor nodes may be dropped from a plane. In such cases, certain probability distributions are used to describe the sensor deployment. Poisson distribution and the Uniform distribution are the most commonly used distributions to model such a deployment [95]. In such a case, by assuming an appropriate distribution it may be possible to derive information about the location of sensor nodes in a sensor field. Also, the location of the nodes may be available from allocation, which is a step performed prior to scheduling. In order to get information about desired phenomenon spatial queries can be issued to sensors based on their location. It may be possible to derive information about the wireless sensor network by getting information from a few preselected nodes instead of querying all the nodes in the network. When a few preselected nodes respond to a query, it is called snapshot querying. In our research, the proxy unicasts request messages to the relevant sensors via

the cluster head. The unicast transmission uses the RTS/CTS/ACK dialogue at the MAC layer and provides reliable transmission. The broadcast transmissions do not employ any of these mechanisms and do not provide the same reliability. In fact in order to achieve reliable broadcast in a WSN, one of the methods is to provide a reliable unicast on each available link [96]. The messages relating to the routing protocol, however broadcast beacons to their neighbors. The individual sensor nodes send the query response messages to the cluster head which finally sends them to the proxy. Such traffic is commonly referred to as convergecast and is assumed by most of the researchers in the field.

Various researchers have talked about disadvantages of flooding and broadcasting the queries in a wireless sensor network. One of the disadvantages is unnecessary energy consumption of nodes that are not expected to respond to the query [97]. Researchers are working on models to reduce flooding in order to save the energy of the resource constrained sensors [98]. Also, in many sensor network applications the sensor nodes sleep for most of the time. They wake up and communicate based on a localized schedule. In such a case, it is reasonable to assume unicast is used because all the nodes within communication distance of a node may not wake up at the same time (otherwise, global synchronization is required) [99].

3.4 Definitions of Some Important Terms Used in this Research

Before describing the proposed scheduling algorithms and allocation algorithms in Section 3.5 and Section 3.6 respectively, the terms application, application request, and job request as used in this research are explained.

3.4.1 Application

An application is a program or software that is meant to collect and/or process data from some or all the sensor nodes. Each application is characterized by the number of sensors required, execution time, and the size of its request and response messages.

3.4.2 Application Request

Users invoke applications on the computing grid. Applications in turn generate requests for the proxy.

3.4.3 Job Request

For each application request that arrives at the proxy, the proxy creates one or more job requests, one for each sensor that is required to participate in the application. The proxy forwards the job requests to the cluster head that finally transmits the job requests to the relevant sensors using a multi-hop communication mechanism. The sensors execute the job request, and create a response message to be transmitted to the cluster head and then finally to the proxy.

3.5 Algorithms for High Performance Scheduling in a WSN

In this section, the scheduling algorithms that are proposed in this research are described. The scheduling algorithm determines the order in which the application requests submitted at the proxy node will be scheduled. The performance of the proposed algorithms is compared with the First Come First Served (FCFS) scheduling algorithm. The FCFS scheduling algorithm is a knowledge free algorithm. It does not use any information about the characteristics of the various applications while scheduling job requests corresponding to the various applications. With the FCFS algorithm, the

application requests are scheduled in the order in which they arrive at the proxy. The job requests corresponding to an application request are sent to the wireless sensor network in the order in which the corresponding application request arrives at the proxy. This research investigates the importance of using knowledge based scheduling as compared to knowledge free scheduling and the gains in performance that can be achieved by using knowledge of application and/or the knowledge of network while making scheduling decisions. Some experiments are performed to determine if knowledge of application or knowledge of network yields any gain in system performance over knowledge free scheduling. Two simple algorithms: Least Number of Sensors First (LNSF) and Least Number of Hops First (LNHF) are proposed and evaluated [100]. The LNSF algorithm uses information only about the number of sensors required by the applications and schedules requests for applications in non-decreasing order of the number of sensors required by the applications. The application indicates the area to be monitored and the phenomenon to be monitored associated with the application. Using this information and the accuracy of measurement required, the proxy can determine the number of sensors required to serve an application request. The LNHF algorithm uses information about network alone while scheduling requests for applications. The requests for applications for which the required sensors are placed closer to the cluster head of a clustered WSN are scheduled first. The LNHF algorithm schedules the application requests in non-decreasing order of the average distance of the sensors required by an application from the cluster head. This algorithm does not use any information about the number of sensors required by the applications. A clustered WSN comprising of 100 sensors and hosting two applications is simulated [100]. For both the applications, the execution time

of job requests corresponding to an application request is assumed to be 5 ms. The data size of request and response messages for both the applications is assumed to be 100 bytes. Similar values of execution time and data size have been used for the other experiments described in detail in Chapter 4. The performance of the algorithms is compared in terms of overall mean response time. Mean Response Time is the mean of the response times of the application requests submitted to the proxy. Response time of an application request is the difference between the times the proxy receives the last response for a job request corresponding to an application request from a sensor and the time when the application request arrives at the proxy.

The performance of LNSF and FCFS is compared. Both the applications vary in their resource requirements. The two applications require 25 and 40 sensor nodes respectively. However, in order to focus on the impact of using the knowledge of sensors in scheduling, the average distance of sensors required by the applications is considered to be the same for both the applications. For such a scenario, performance comparison of LNSF and FCFS is shown in Figure 3-5. The overall mean response time obtained with LNSF is less than the overall mean response time obtained with FCFS. Clearly, using the knowledge of application brings about improvement in system performance.

The performance of LNHF is also compared to FCFS. Two applications requiring the same number of sensors but varying in their average distance of the required sensors from the cluster head are considered [100]. The number of sensors required by both the applications is fixed at 25. The sensors required for Application 1 are placed one hop away from the cluster head and the sensors required for Application 2 are placed 1-4 hops away from the cluster head. This experiment focuses on determining the usefulness of

using the knowledge of network in scheduling. The performances of LNHF and FCFS for such a scenario are shown in Figure 3-6.

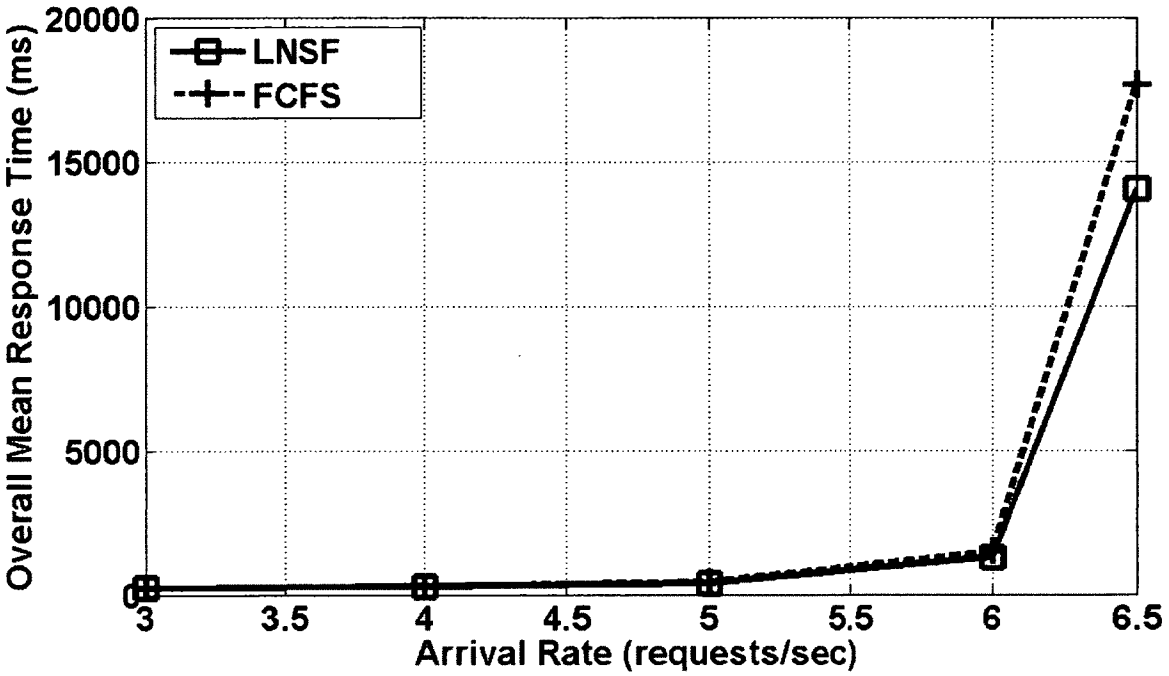


Figure 3-5: Performance Comparison of LNSF and FCFS Scheduling

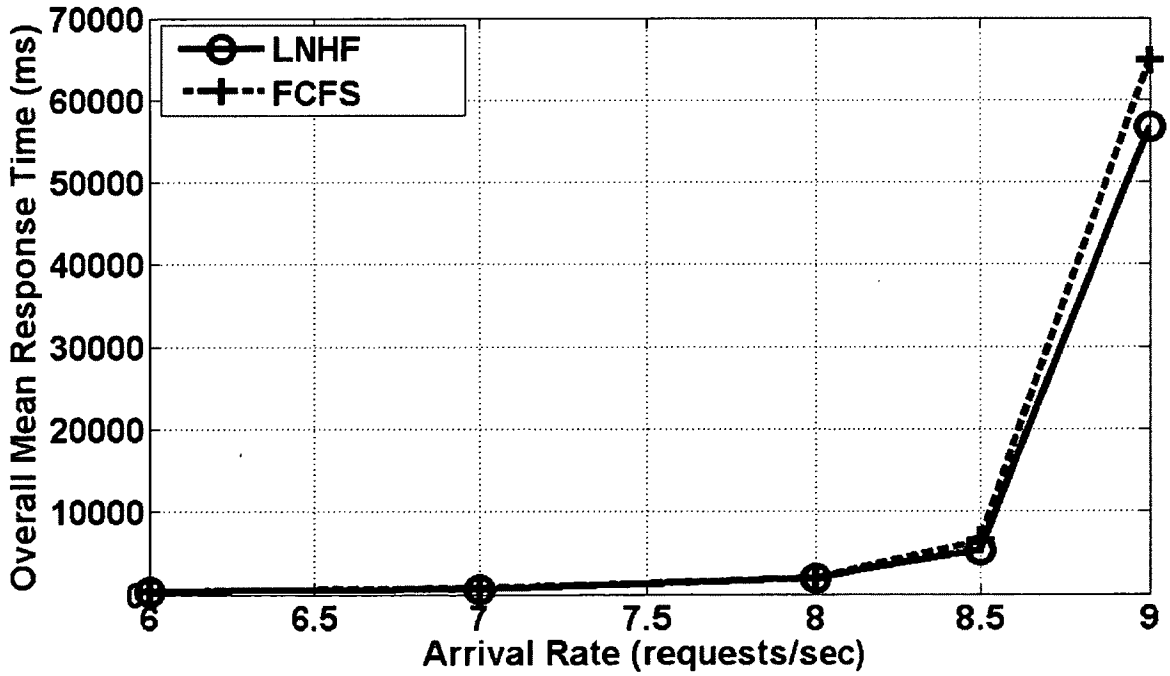


Figure 3-6: Performance Comparison of LNHF and FCFS Scheduling

The LNHF algorithm demonstrates a better performance than the FCFS algorithm and provides a better overall mean response time. Thus, using the knowledge of network also seems to be important in making a scheduling decision.

The results described earlier motivate a detailed investigation of scheduling algorithms that combine the knowledge of application and networks in WSNs hosting multiple applications with varying resource requirements and location of sensors. Such scheduling algorithms are discussed next.

The scheduling algorithms proposed in this thesis are shown in Figure 3-7.

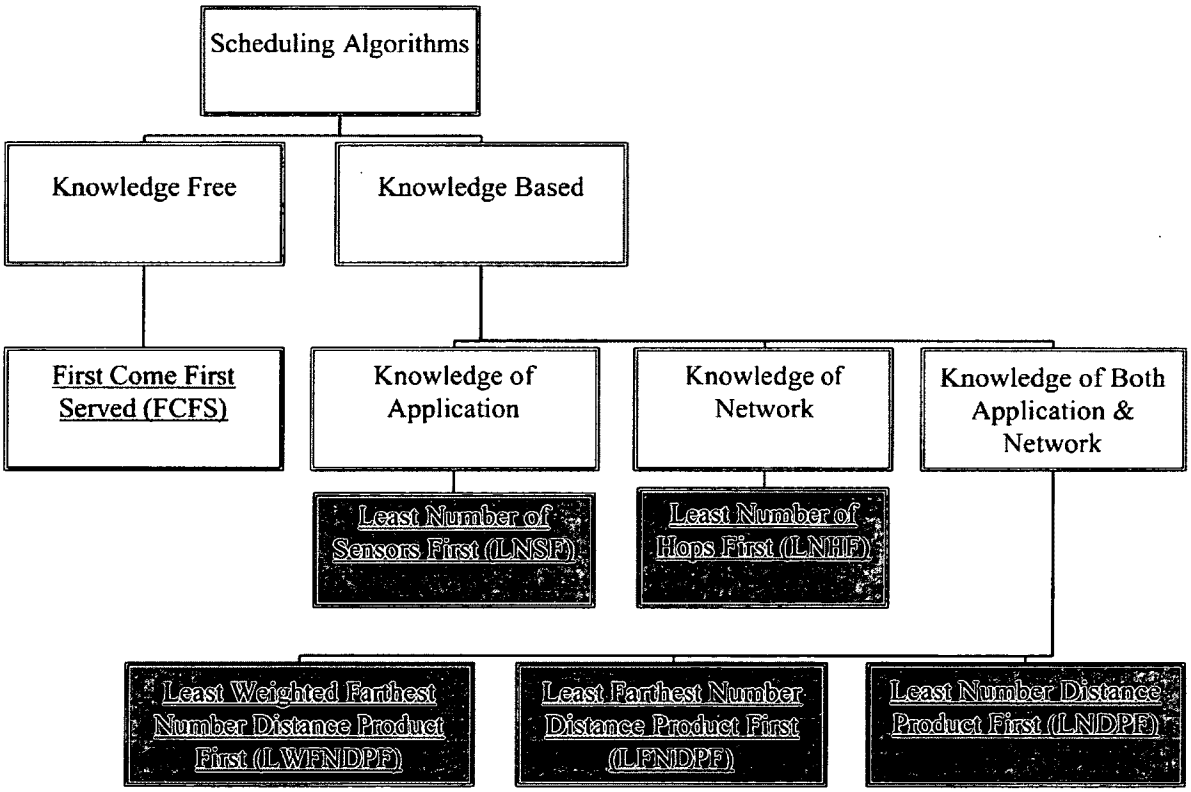


Figure 3-7: Scheduling Algorithms

Two classes of algorithms are considered: knowledge based and knowledge free. FCFS is a knowledge free algorithm that schedules the requests in the order in which they

arrive. Knowledge based algorithms either use knowledge of applications or network or both the knowledge of application and network while making a scheduling decision.

The Least Number of Sensors First (LNSF) algorithm takes scheduling decisions based only on the number of sensors required by the applications and the Least Number of Hops First (LNHF) algorithm takes scheduling decisions based on the average distance of the sensors required by the applications from the cluster head. These algorithms were described earlier in this section. The LNSF and the LNHF algorithms have been proven to be suitable for networks hosting a single class of applications that either use a similar number of sensors or for WSNs hosting applications with similar average distances of required sensors from the cluster head [100]. For WSNs hosting applications with different characteristics and network related requirements, whether or not combining both the knowledge about the resource requirements of an application and the network topology in scheduling leads to a performance improvement is an important question. The Least Number Distance Product First (LNDPF) algorithm, Least Farthest Number Distance Product First (LFNDPF) algorithm, and the Least Weighted Farthest Number Distance Product First (LWFNDPF) algorithm combine the knowledge of the network and application while making a scheduling decision. These algorithms are described in detail later in this section.

In this research, the scheduling algorithms are deployed at the proxy. As the sensor nodes are resource constrained and have limited energy, this conserves the energy consumption at the sensor nodes due to the deployment of the algorithms at the proxy. It has been shown that comparable performance can be achieved when the scheduling algorithms are deployed at the proxy instead of the sensor nodes [100]. The sensor nodes

serve the job requests corresponding to the applications in the order in which they receive them. Job requests corresponding to applications that have a higher priority of scheduling as determined by the scheduling algorithm have a higher priority of transmission at the MAC layer of the cluster head. A priority queue is maintained between the link layer and the MAC layer. The link layer and the MAC layer form the data link layer corresponding to the OSI Model [101]. Different level of service can be provided to the packets by defining the type of service in the packet structure. The priority queue maintains separate queues based on the number of priority levels of the packets. When the MAC layer finishes the current transmission, it sends a null message back to the queue so that the queue sends in the next packet. The queue sends packets in accordance with the packet priority that is determined by the scheduling algorithm used. The corresponding response messages also have a higher priority of transmission at the sensor nodes.

As discussed, several knowledge based scheduling algorithms have been proposed in this research. The performance of the proposed algorithms is compared with that of the knowledge free First Come First Served scheduling algorithm. The algorithms are described in detail next.

3.5.1 Least Number Distance Product First (LNDPF)

Applications have varying requirements in terms of the number of sensors required by the applications and also the location of the sensors required by the applications with respect to the cluster head. The Least Number Distance Product First algorithm considers all the sensors required by an application while making a scheduling decision. Number Distance Product (NDP) for an application is the product of the number

of sensors required by an application and the average of distance of all the sensors required by an application from the cluster head. The distance of a sensor from the cluster head is measured in terms of the number of hops between the sensor node and the cluster head. With the LNDPF algorithm, the requests from the applications are scheduled in the order of non-decreasing NDP. The pseudo code for the LNDPF algorithm is provided in Appendix A.1.

3.5.2 Least Farthest Number Distance Product First (LFNDPF)

Farthest distance is the distance of the farthest sensors required by an application from the cluster head. Farthest Number Distance Product (FNDP) for an application is the product of number of sensors required for a particular application located at the farthest distance from the cluster head and the farthest distance. The time required by a request to reach the relevant sensor and also the time required by a response to reach the cluster head from the sensor is proportional to the distance of the sensor from the cluster head. The responses from the sensors that are located farthest away from the cluster head are expected to require more time in reaching the cluster head as compared to responses from the sensors that are located closer to the cluster head. The response for an application request is complete when responses for all the job requests corresponding to the application request are received back by the proxy. Therefore, the responses from the farthest sensors will be instrumental in determining the mean response time of an application request and hence the overall mean response time. As the sensors comprising the wireless sensor network share the wireless medium for transmission of request and response messages, the time required for all the responses from the sensors located at

farthest distance will also be dependent on the number of sensors located farthest distance away from the cluster head. The Least Farthest Number Distance Product First algorithm uses both these attributes and schedules the requests from the application that has the least FNDP first. The pseudo code for the LFNDPF algorithm is provided in Appendix A.2.

3.5.3 Least Weighted Farthest Number Distance Product First (LWFNDPF)

In a WSN, messages to and from the sensors that are located farther away from the cluster head experience greater delays as compared to sensors that are located closer to the cluster head. As the sensors share the wireless medium, delays are experienced while transmitting a message at each hop due to interference experienced from neighboring sensors. The greater the hop distance of a sensor from the cluster head, greater is the delay incurred by the message to and from the sensor. Various studies have shown a decrease in throughput over multi-hop routes as compared to single hop routes [102]. In [102], the authors show that a two hop route only has half the capacity of a one hop route and a three hop route has a capacity of one third of a single hop route. The achievable capacity of a wireless network is largely affected by the network size, traffic volume, and detailed radio interactions [103]. The messages that are transmitted to and from the farthest sensor nodes in the network experience largest delays as they experience delays while being transmitted and received at each intermediate hop due to the interference received from other nodes [102][103]. Based on this rationale, this algorithm assigns a weight to each hop distance. The higher hop distances are allocated a higher weight as compared to lower hop distances. The FNDP for each application is

multiplied by the appropriate weight at that particular hop to account for the greater interference experienced by messages to and from the sensors located at a higher hop distance from the cluster head. This gives the Weighted Farthest Number Distance Product (WFNDP). With the Least Weighted Farthest Number Distance Product First algorithm, the requests from the various applications are scheduled in order of non-decreasing WFNDP. The weights allocated are 0.05, 0.2, 0.7, and 1 at distances of 1 hop, 2 hops, 3 hops, and 4 hops from the cluster head respectively. Various weight values are experimented with. The chosen weights led to superior system performance for most of the configurations experimented with. To check the sensitivity of the performance of the LWFNDPF algorithm on the allocated weight values, the weight values for each hop distance are randomly modified by 25 to 50 % of the allocated values. Several sets of weight values are obtained while the following condition is satisfied: Higher hop distances have higher weights as compared to lower hop distances. The sets of weight values experimented with are shown in Table 3-1. It is observed that the relative performances of the scheduling algorithms are not sensitive to the allocated weight values. The pseudo code for the LWFNDPF algorithm is provided in Appendix A.3.

Table 3-1: Sets of Weight Values Experimented With for the LWFNDPF Algorithm

Set No.	Hop 1	Hop 2	Hop 3	Hop 4
1	0.05	0.2	0.7	1
2	0.02	0.08	0.28	0.38
3	0.019	0.096	0.21	0.46

For all the proposed algorithms, if some of the applications have the same value of NDP, FNDP, or WFNDP, the tie is broken using the FCFS paradigm.

3.6 Algorithms for High Performance Allocation in a WSN

Allocation is the process of selection of sensor nodes from the available nodes in the WSN that would execute the job requests corresponding to a specific application request. The main aim of an allocation algorithm is to increase the lifetime of the wireless sensor network. The lifetime of the WSN can be increased by increasing the lifetime of the individual sensors. Allocation of sensor nodes to job requests corresponding to the various applications may either be static or dynamic. In case of static allocation, the sensor nodes are allocated to the various applications at the beginning and this allocation is not changed for the entire lifetime of the wireless sensor network. In case of dynamic allocation, the sensor nodes are allocated to the job requests corresponding to an application every time an application request arrives at the proxy. Both, static allocation algorithms and dynamic allocation algorithms are proposed in this research. As described in Section 3.2.1, the major energy consuming components in a sensor are the CPU component and the radio component. The battery component provides the energy to the radio component and the CPU component. Various algorithms that use no information or varying degree of information about the energy consumption at the CPU component, energy consumption at the radio component, and the available energy at the battery component have been proposed. The various allocation algorithms proposed in this research are shown in Figure 3-8.

Figure 3-8 presents the various classes to which an algorithm belongs. The intermediate nodes in the tree correspond to the algorithm class whereas each leaf node corresponds to an algorithm (with its name underlined). The algorithms can be divided into two classes: static and dynamic. With a static allocation, the sensor nodes are

mapped to applications *a priori*. For all requests for a given application the same nodes are used. For a dynamic allocation algorithm sensor allocation is done for each request arrival. Thus, different requests for the same application may be served by different sensor nodes.

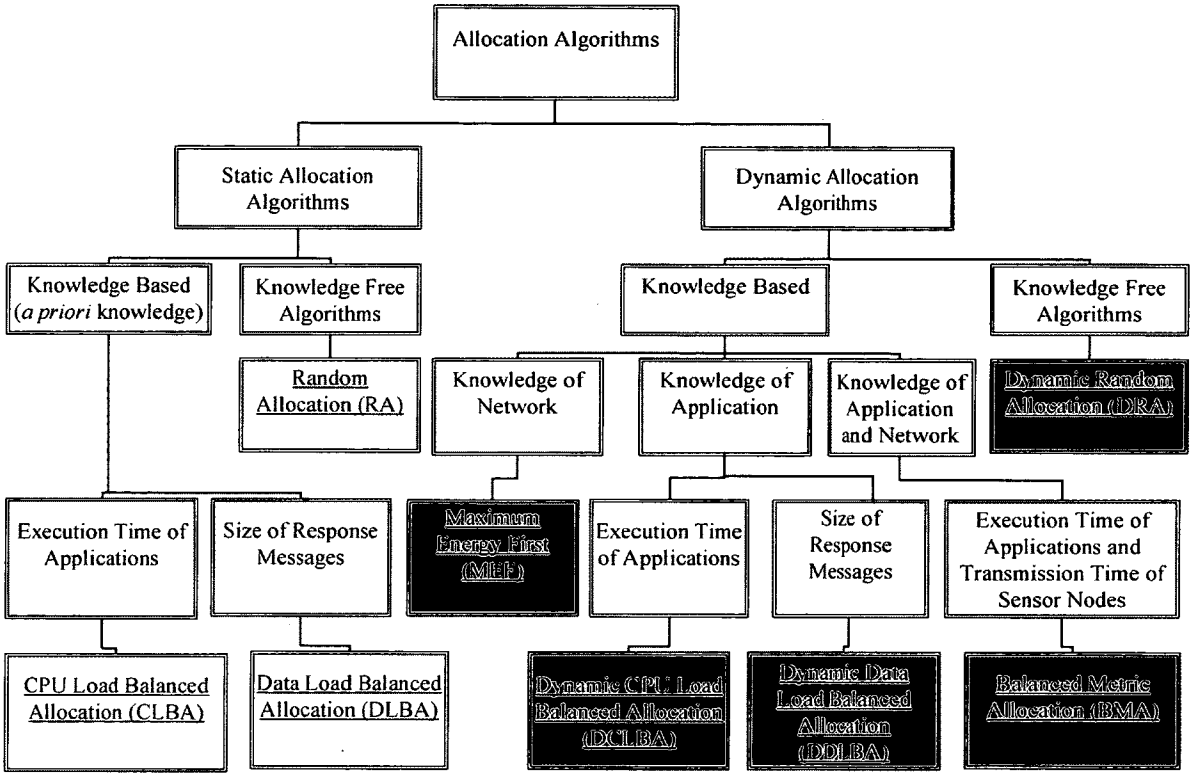


Figure 3-8: Allocation Algorithms

In case of static allocation algorithms, both knowledge free and knowledge based algorithms that are based on *a priori* knowledge are proposed. In the category of knowledge free algorithms a Random Allocation (RA) algorithm is proposed. RA allocates sensor nodes to requests for applications at random. Each sensor node has an equal probability of being selected. In the category of knowledge based algorithms, algorithms that use information about the execution time of applications or the size of messages are proposed. CPU Load Balanced Allocation (CLBA) makes use of

information about the execution time of applications in order to balance the energy consumption at the CPU component of the various sensor nodes. Data Load Balanced Allocation (DLBA) focuses on balancing the load at the radio component due to the transmission of response messages corresponding to requests processed by a node amongst the various sensor nodes.

In case of dynamic allocation algorithms also, both knowledge free and knowledge based algorithms are proposed. The Dynamic Random Allocation (DRA) is a knowledge free algorithm that allocates sensor nodes at random upon arrival of every request. The knowledge based algorithms either use knowledge of application and/or the knowledge of the network while making an allocation decision. Maximum Energy First (MEF) uses knowledge of energy level of sensor nodes at all times and allocates sensor nodes based on their energy levels. The Dynamic CPU Load Balanced Allocation (DCLBA) algorithm and the Dynamic Data Load Balanced Allocation (DDLBA) algorithm focus on balancing the energy consumption at the CPU component and the radio component respectively amongst the sensor nodes. The Balanced Metric Allocation (BMA) algorithm aims to balance the total energy consumption amongst the sensor nodes. The allocation algorithms proposed in this research are described in detail next.

3.6.1 Static Allocation Algorithms

With the static allocation algorithms, the sensor nodes are allocated in the beginning to the various applications supported by the wireless sensor network. The allocation of the sensor nodes to the application does not change for every arrival of an

application request at the proxy. The static allocation algorithms proposed in this research are described next.

3.6.1.1 Random Allocation (RA)

This algorithm does not use any information about the characteristics of the application or the network while making an allocation decision. The sensor nodes are allocated to the various applications at random. Each sensor node has the same probability of being selected. On each arrival of an application request at the proxy, the job requests corresponding to the application request are submitted to the sensor nodes that have been allocated to that application. The pseudo code for the RA algorithm is provided in Appendix B.1.1.

3.6.1.2 CPU Load Balanced Allocation (CLBA)

The CPU is one of the major power consumers in a sensor node. The CPU component draws more current in active state as compared to the current drawn in the idle state. The time the CPU component spends in the active state depends on the execution time of the application. The CLBA algorithm focuses on balancing the energy consumption due to the CPU component amongst the various sensor nodes. By using such an approach, difference between the sums of execution times of requests submitted to the sensor nodes is minimized. The CLBA algorithm performs static allocation by balancing the sum of execution time of applications to which the sensor nodes are allocated. Balancing means "to equalize". The term balancing has the same meaning in the context of other algorithms as well. The pseudo code for the CLBA algorithm is provided in Appendix B.1.2.

3.6.1.3 Data Load Balanced Allocation (DLBA)

Apart from the CPU, one of the major power consumers in a sensor node is the radio component. The radio component consumes more energy while transmitting as compared to when it is in idle mode or receive mode. The default mode of the radio component is assumed to be the receive mode. The radio component expends additional energy while transmitting response messages corresponding to a request processed by the sensor node. This algorithm does not consider any messages that may be relayed by a sensor node as that information is not known *a priori*. The energy spent while transmitting a response message is proportional to the size of the response message. This algorithm focuses on balancing the energy consumed while transmitting the response messages corresponding to the requests processed by the sensor node. The DLBA algorithm tries to balance the sum of data size of response messages of applications to which the sensor node is allocated amongst all sensor nodes. The pseudo code for the DLBA algorithm is provided in Appendix B.1.3.

3.6.2 Dynamic Allocation Algorithms

With the dynamic allocation algorithms, the job requests corresponding to an application request are allocated dynamically when the application request arrives at the proxy. The allocation can change for one request to another even for the same application.

3.6.2.1 Dynamic Random Allocation (DRA)

With this algorithm, the sensors required to execute the job requests corresponding to an application request are selected at random. This algorithm does not

use any information about the state of the wireless sensor network or the characteristics of the application while making an allocation decision. The pseudo code for the DRA algorithm is provided in Appendix B.2.1.

3.6.2.2 Dynamic CPU Load Balanced Allocation (DCLBA)

Similar to the CLBA algorithm, the DCLBA algorithm aims to balance the energy consumption at the CPU component of the various sensor nodes. However, with this algorithm each application request is allocated dynamically upon arrival at the proxy. For each sensor node, the proxy maintains information about the sum of execution time of job requests that have been allocated to the sensor nodes. The job requests are allocated to sensor nodes that have lower value of sum of execution times of job requests that have been allocated to a sensor node. The pseudo code for the DCLBA algorithm is provided in Appendix B.2.2.

3.6.2.3 Dynamic Data Load Balanced Allocation (DDLBA)

This algorithm focuses on balancing the energy expended by the radio component amongst the sensor nodes. This algorithm allocates job requests corresponding to an application request dynamically upon arrival of an application request at the proxy. A sensor node that has been allocated to a job request needs to transmit the response after the request has been processed. Transmission of response messages from the sensor node expends energy. This algorithm aims to balance the energy consumption at all sensor nodes due to transmission of response messages which may also be referred to as the data load on the sensor nodes. The energy consumed while transmitting a response message is directly proportional to the size of the response message. Applications may have varying

requirements in terms of the size of the response message to be transmitted. With this algorithm, for each sensor node, the proxy maintains information about the sum of the size of the various response messages that have been transmitted by the sensor nodes. The sensor nodes with lower value of sum of the size of the various response messages that have been transmitted by the sensor nodes are allocated first. The pseudo code for the DDLBA algorithm is provided in Appendix B.2.3.

3.6.2.4 Balanced Metric Allocation (BMA)

The CPU component and the radio component are the major power consumers in a sensor node. The Balanced Metric Allocation algorithm aims to balance the energy consumption amongst all the sensor nodes, both due to the CPU component and the radio component.

For the CPU component, the additional energy consumed due to the allocation of job requests to a sensor node is considered. For example, if the current drawn by the CPU in idle mode is $I_{CPUIdle}$ and the current drawn by the CPU in the active state is $I_{CPUActive}$, then the additional energy consumed by the sensor node at the CPU component due to the allocation of job requests to the sensor node (E_{CPU}) is equal to the product of the battery voltage, difference in the current drawn by the CPU in the idle state and the active state and the sum of the execution times of the job requests allocated to the sensor node ($Execution\_Time_{Sum}$).

$$E_{CPU} = V_{Battery} * (I_{CPUActive} - I_{CPUIdle}) * Execution\_Time_{Sum}$$

For the radio component, additional energy is consumed while transmitting a message. The message may be a sensor response corresponding to the request processed

by the sensor node or any other message being relayed by the sensor node. If the current drawn by the radio in idle/receive mode is I_R and the current drawn by the radio in the transmitting state is I_T, then the additional energy consumed by the sensor node due to the transmission of the messages is equal to the product of the battery voltage, difference in the current drawn by the radio in the transmitting state and the idle state and the and the sum of the transmission times of the messages transmitted by the sensor node ($Transmission\_Time_{Sum}$).

$$E_{Radio} = V_{Battery} * (I_T - I_R) * Transmission\_Time_{Sum}$$

The time required to transmit a message is proportional to the size of the message and may be computed using the transmission bandwidth.

A metric is defined as the sum of E_{CPU} and E_{Radio}. The Balanced Metric Allocation algorithm tries to balance the energy consumption amongst sensor nodes by minimizing the difference between the sum of E_{CPU} and E_{Radio} amongst the sensor nodes. Sensor nodes are allocated to job requests in increasing order of this metric. The pseudo code for the BMA algorithm is provided in Appendix B.2.4.

3.6.2.5 Maximum Energy First (MEF)

In case of the MEF algorithm, the sensor nodes that have the highest available energy are selected from amongst the sensor nodes in the WSN for execution of the job requests corresponding to an application request. The rationale behind the algorithm is to balance the energy level amongst the various sensor nodes and to increase the network lifetime of the wireless sensor network by delaying the time when the energy level of any sensor node falls below a predefined threshold. This algorithm assumes complete

knowledge about the energy level of all sensor nodes, and allocates the job requests corresponding to an application request to the sensor nodes that have higher amount of available energy. The energy level of sensor nodes may be piggybacked on the messages sent by the sensor nodes to the proxy. The pseudo code for the MEF algorithm is provided in Appendix B.2.5.

Experiments are performed to compare the performances of the scheduling algorithms and the allocation algorithms proposed in this research. The performance measures that have been used to measure the performance of the scheduling and allocation algorithms are described in the next section.

3.7 Performance Measures

The performance indicators that have been used to assess the performance of the scheduling and allocation algorithms introduced in this research are described here.

3.7.1 Mean Response Time (R_t)

The main aim of any scheduling algorithm is to reduce the overall mean response time to the users of the applications. *Mean Response Time* is the mean of the response times of the application requests submitted to the proxy. Response time of an application request is the difference between the times the proxy receives the last response for a job corresponding to an application request from a sensor and the time when the application request arrives at the proxy. It is calculated by taking the sum of the response times of all the application requests submitted to the proxy divided by the total number of submitted requests.

3.7.2 Minimum Energy

As discussed earlier, one of the aims of any allocation algorithm is to increase the lifetime of the wireless sensor network. The lifetime of the WSN is determined by the lifetime of the individual sensors comprising the wireless sensor network. A sensor node may cease to perform when the energy of a sensor node falls below a threshold. *Minimum Energy* at a sensor node at the end of the simulation period is used as a performance metric to determine the performance of the allocation algorithms proposed in this thesis.

3.7.3 Network Lifetime

The performance of allocation algorithms may also be measured in terms of network lifetime. *Network Lifetime* is defined as the time when the energy of any one sensor node falls below a pre-defined energy value.

3.7.4 Rate of Energy Drain

Rate of Energy Drain at a sensor node is defined as the difference between the initial energy and the final energy at the end of the simulation period at a sensor node divided by the given time period.

3.8 Simulation Parameters

This section describes the system and workload parameters to be used in the simulation experiments.

3.8.1 System Parameters

The system parameters, MAC and physical layer parameters of interest are presented next.

Number of Sensors (N_s)

This is the total number of sensors in the wireless sensor network.

MAC_RTSThreshold

This parameter determines the threshold data size for using the RTS/CTS handshaking mechanism.

MAC_ShortRetryLimit

This is the maximum number of retransmissions allowed when the size of the frame is less than the MAC_RTSThreshold.

MAC_LongRetryLimit

This is the maximum number of retransmissions allowed when the size of the frame is greater than the MAC_RTSThreshold.

MAC_FragmentationThreshold

This is the maximum size of the frame that can be transmitted without fragmentation.

MAC_MaxTransmitMSDULifetime

This parameter specifies the maximum time a frame to be transmitted may remain in the MAC layer before being sent to the channel.

MAC_MaxReceiveLifeTime

This parameter specifies the maximum time a received frame may remain in the MAC layer before being retransmitted on the channel.

CWMin

This parameter specifies the minimum size of the congestion window.

CWMax

This parameter specifies the maximum size of the congestion window.

Transmitter Power

This is the transmission power of the antenna.

Rx Threshold

This parameter specifies a power threshold. If a packet reaches a node with a power level that is below Rx Threshold, the node will receive the packet with error, but will consider the channel to be busy for duration of time equal to the transmission time of the packet. A free space path loss model has been considered in this research [88].

CS Threshold

This parameter specifies a power threshold. If a packet reaches a node with a power level that is below CS Threshold, the node will not be able to sense the transmission and will perceive the channel to be idle.

CP Threshold

This parameter specifies a power threshold. If a node is receiving a packet and another newly transmitted packet reaches the node with a power that is below the CP Threshold, the newly transmitted packet will not interfere with the reception of the packet being received. Otherwise, the packets will collide and the node will not be able to read either of the packets.

Bandwidth

This is the bandwidth of the wireless links connecting the sensor nodes and is measured in Mbps.

Propagation delay

After a bit of data is pushed on the link, it has to move from the transmitting entity to the receiving entity. Propagation delay is the time required by the bit to propagate from the transmitting entity to the receiving entity. Since the distance between the sensor nodes is very small, propagation delay is very small as compared to the time required to transmit the data over the network. Due to its small value, the propagation delay has been ignored in this research. Similar assumptions have been used by other researchers [88].

3.8.2 Workload Parameters

This section describes the workload parameters to be used in the simulation experiments.

Arrival Rate (λ)

It is the number of application requests generated per second for the entire system.

$$\lambda = \sum \lambda_{appi}$$

λ_{appi} denotes the arrival rate of requests for application from users of applications. Requests for various applications from users of applications have an equal probability of arrival denoted by p.

Number of applications ($N_{applications}$)

This is the total number of applications that share the wireless sensor network.

Data Size

This is the size of the job requests and the sensor response messages (in bytes) that are transmitted over the network. The size of the job requests corresponding to an

application request and the size of the sensor response messages are considered to be the same for the default set of workload parameters (see Table 4-1).

Number of sensors required by Application i (S_{appi})

This is the number of sensors required by application i.

Execution Time of Application requests (E_{appi})

This is the execution time (in milliseconds) of a job request belonging to application request of application i. The sensor nodes are assumed to be homogeneous in this research. Hence, the execution time of job requests corresponding to an application request on all sensor nodes is equal.

The simulation experiments that have been performed to get insights into the performance of the scheduling and allocation algorithms are described in Chapter 4 and Chapter 5 respectively.

Chapter 4: Performance of Scheduling Algorithms

This chapter describes the simulation experiments that have been performed to study the performance of the proposed scheduling algorithms. Effect of various system and workload parameters on the performance of the proposed scheduling algorithms is presented.

The questions that motivate the performance analysis include the following:

- Which characteristics of the system and/or application need to be used in scheduling?
- How do the scheduling algorithms proposed in this research that use various degrees of knowledge about characteristics of application and network perform?
- Which system and workload parameters have a significant effect on the performance of the proposed scheduling algorithms?

The simulation model used in this research has been described in Section 3.3. In order to study the effect of a given parameter on performance, a factor at a time approach has been used in our simulation experiments. The parameter of interest is varied while all the other parameters are held at their default levels (see Table 4-1 and Table 4-2). Table 4-1 lists the parameters pertaining to the system, the workload, and the application layer. Table 4-2 lists the default values of the key parameters pertaining to the physical and the MAC layer. Other researchers have used similar values [104] [105] [88]. A default topology that defines the location of the sensors required by various applications is given

in Table 4-3. The scheduling algorithms analysed in this chapter are summarized in Table 4-4.

Table 4-1: Default Values of System and Workload Parameters

Parameter	Value
p	0.2
λ_{app1}, λ_{app2}, λ_{app3}, λ_{app4}, λ_{app5}	$p\lambda$ (0.2λ)
N_s	125
$N_{applications}$	5
S_{app1}	15
S_{app2}	20
S_{app3}	25
S_{app4}	30
S_{app5}	40
Degree of Sharing	10
Data Size	100 bytes
E_{app1}, E_{app2}, E_{app3}, E_{app4}, E_{app5}	5 ms

Table 4-2: Default Values of Physical and MAC Layer Parameters

Parameter	Value
MAC_RTSThreshold	3000 bytes
MAC_ShortRetryLimit	7 retransmissions
MAC_LongRetryLimit	4 retransmissions
MAC_FragmentationThreshold	2346 bytes
MAC_MaxTransmitMSDULifetime	512 time units
MAC_MaxReceiveLifeTime	512 time units
CWMin	31
CWMax	1023
Bandwidth	2 Mbps
Transmitter Power	0.2818 W
Rx Threshold	1.981*10-8 W
CS Threshold	$0.035225*10^{-8}$ W
CP Threshold	10 db
Path Loss Model	Free Space Model

Table 4-3: Default Topology

Application	Sensors at 1 hop distance	Sensors at 2 hop distance	Sensors at 3 hop distance	Sensors at 4 hop distance	NDP	FNDP	WFNDP
Application 1	10			5	30	20	20
Application 2	10			10	50	40	40
Application 3	10		15		55	45	31.5
Application 4	10	15	5		55	15	10.5
Application 5	25	10	5		60	15	10.5

Table 4-4: Scheduling Algorithms

Scheduling Algorithm	Acronym	Description	Reference
First Come First Served	FCFS	Schedules requests for applications in the order in which they arrive.	Section 3.5
Least Number Distance Product First	LNDPF	Schedules requests in non-decreasing order of Number Distance Product for an application.	Section 3.5.1
Least Farthest Number Distance Product First	LFNDPF	Schedules requests in non-decreasing order of Farthest Number Distance Product for an application.	Section 3.5.2
Least Weighted Farthest Number Distance Product First	LWFNDPF	Schedules requests in non-decreasing order of Weighted Farthest Number Distance Product for an application.	Section 3.5.3

The simulation experiments that have been conducted to study the effect of various system and workload parameters on the performance of the proposed scheduling algorithms are described next. The experiments were run long enough and repeated multiple times to obtain sufficiently small confidence intervals for the average values.

For the experiments presented next, confidence intervals of ±5% (or less) for overall mean response time were obtained at a confidence level of 95%.

A part of the results presented in this chapter are included in [106].

4.1 Effect of Arrival Rate on the Performance of Algorithms

The system and workload parameters are held at the default values (given in Table 4-1) and the arrival rate is varied. The topology is the default topology as described in Table 4-3. The performance of various algorithms is shown in Figure 4-1. The LNDPF algorithm provides the worst performance and the LWFNDPF algorithm provides the best performance. The LNDPF algorithm makes a scheduling decision based on all the sensors required by an application. The LFNDPF and the LWFNDPF algorithms make a scheduling decision based on the sensors required by an application that are located farthest away from the cluster head. With the LNDPF algorithm, the applications that require a large number of sensors receive a lower priority of scheduling even though the sensors may be placed closer to the cluster head (Application 4 and Application 5 in this case). LNDPF gives a higher priority to requests from applications that have lower values of NDP due to the smaller number of required sensors, even though some of the required sensors are located farther away from the cluster head as compared to other applications. The LFNDPF and the LWFNDPF algorithms give a higher priority to requests for applications for which the farthest sensors are placed closer to the cluster head. The LWFNDPF algorithm provides the best performance. The results indicate that farthest sensors required by an application play an important role in determining the response time of an application and hence the overall mean response time. The difference

between the performances of the LFNDPF and the LWFNDPF algorithm may be attributed to the difference in the priority allocated to application requests based on the different weights allocated to each hop distance by the LWFNDPF algorithm.

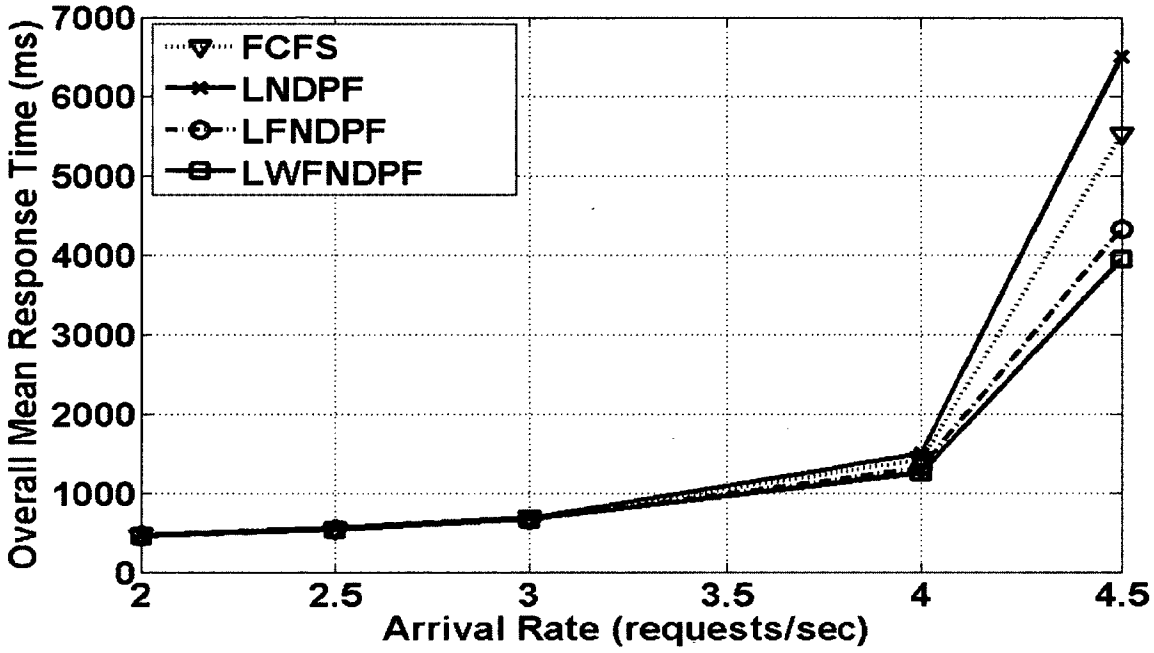


Figure 4-1: Effect of Arrival Rate on the Performance of Algorithms

4.2 Effect of Data Size on the Performance of Algorithms

Keeping all the system and workload parameters at their default values, the data size of the request messages and the sensor response messages is varied. The request messages and sensor response messages are assumed to be of the same size for all applications. The size of the messages is varied and takes the values 50 bytes, 100 bytes, and 150 bytes. Similar values for size of messages are considered by other researchers [104]. For a given arrival rate, the performance of various algorithms for different data sizes is shown in Figure 4-2. The LWFNDPF algorithm demonstrates the best performance. The difference in the relative performances of the algorithms increases with

the increase in data size. This may be attributed to the fact that the transmission time of messages is directly proportional to the size of the messages. This causes greater delays to messages that are waiting to be transmitted. Also, the request messages and the response messages will experience delays while being transmitted at each hop. Therefore the LWFNDPF algorithm that favors requests from applications that require a lower number of sensors placed farther away from the cluster head performs even better for larger data sizes.

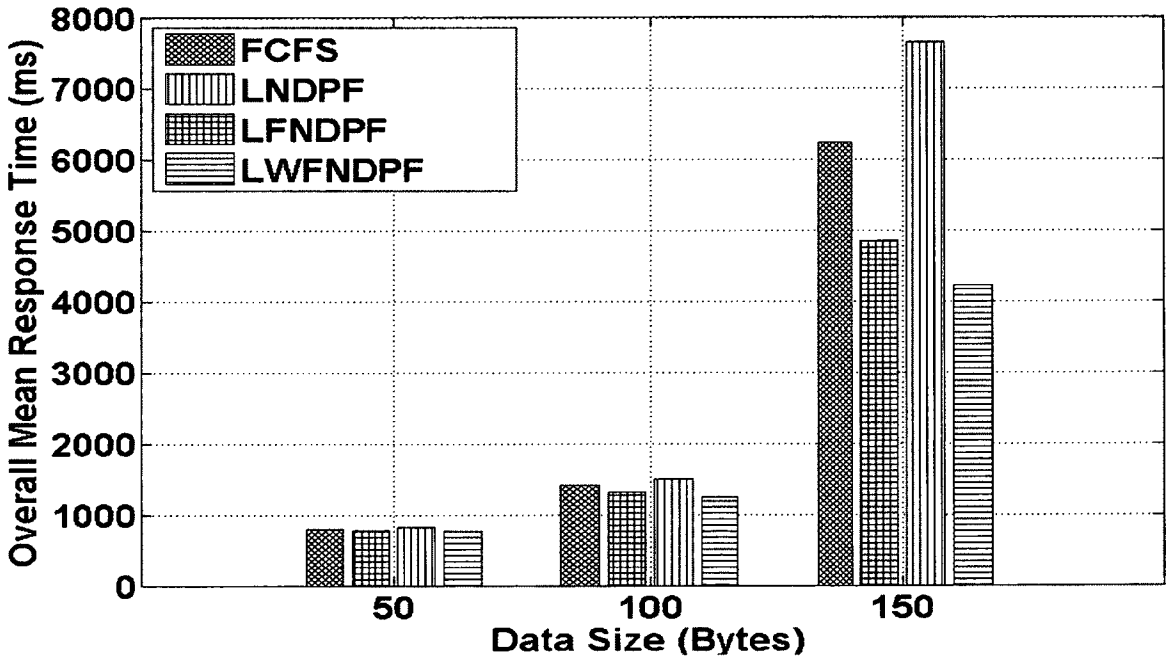


Figure 4-2: Effect of Data Size on the Performance of Algorithms

In the previous experiment, the size of request and response messages for all applications is considered the same. Another experiment is performed to check the sensitivity of the performance of the proposed scheduling algorithms to the variability in size for the request and response messages. The size of the request messages from various applications is uniformly distributed between 80 and 120 bytes. The size of the

response messages for various applications is uniformly distributed between 100 bytes and 150 bytes. The relative performance of the proposed algorithms for a given arrival rate is shown in Figure 4-3. By favoring the request and response messages from applications that have a lower number of sensors placed farther away from the cluster head, the LWFNDPF algorithm demonstrates the best performance. LNDPF demonstrates the worst performance. The performance of other algorithms is close to that of the LWFNDPF algorithm.

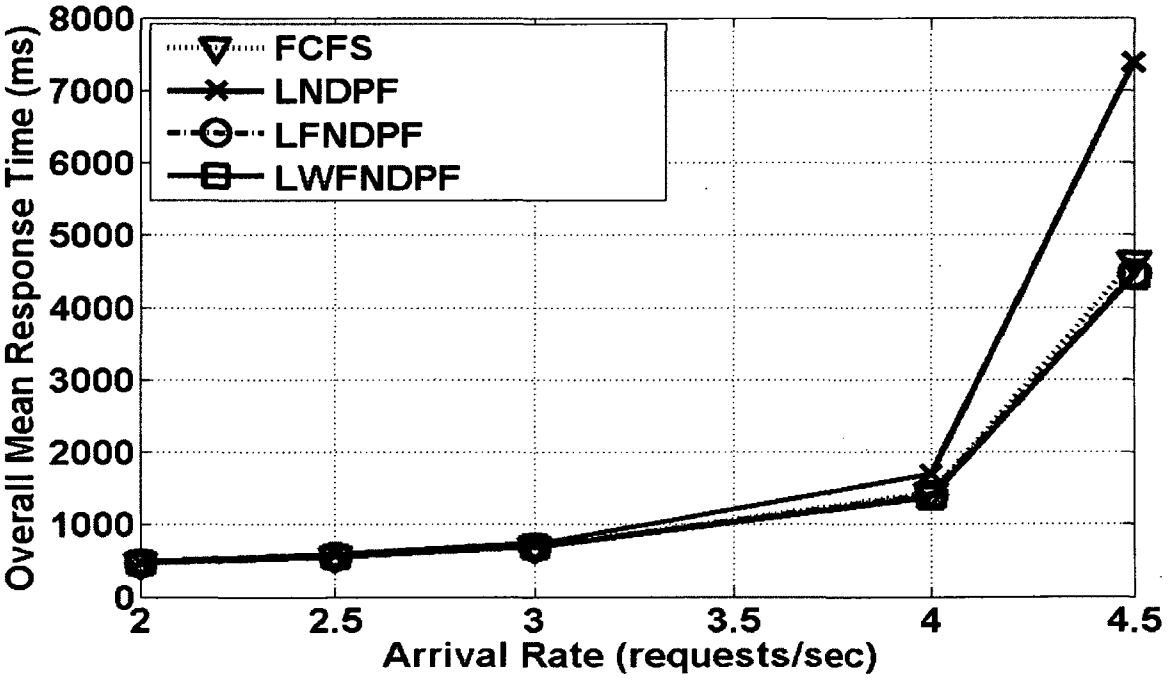


Figure 4-3: Effect of Varying Data Size on the Performance of Algorithms

4.3 Effect of Number of Applications on the Performance of Algorithms

The effect of number of applications on the performance of the scheduling algorithms is studied. The number of applications in the default configuration is five. For one of the configurations, the number of applications is set to two and Application 2 and Application 3 from the set of the five default applications are considered. Based on the

default topology (given in Table 4-3), Application 3 has a higher value of NDP and FNDP than Application 2 whereas Application 3 has a smaller value of Weighted Farthest Number Distance Product than Application 2. This configuration is particularly important to validate the observation for allocating a higher weight to higher hop distances. In another configuration, three applications are considered from the set of the five default applications: Application 1, Application 2, and Application 4. Based on the default topology, Application 4 has the highest NDP but the lowest FNDP from amongst the three applications. This configuration is particularly selected to validate the observation that the sensors placed farthest away from the cluster head play the most important role in determining the overall mean response time. In each configuration, the arrival rate is equally split amongst the applications considered. For a given arrival rate, the performance of the algorithms is shown in Figure 4-4.

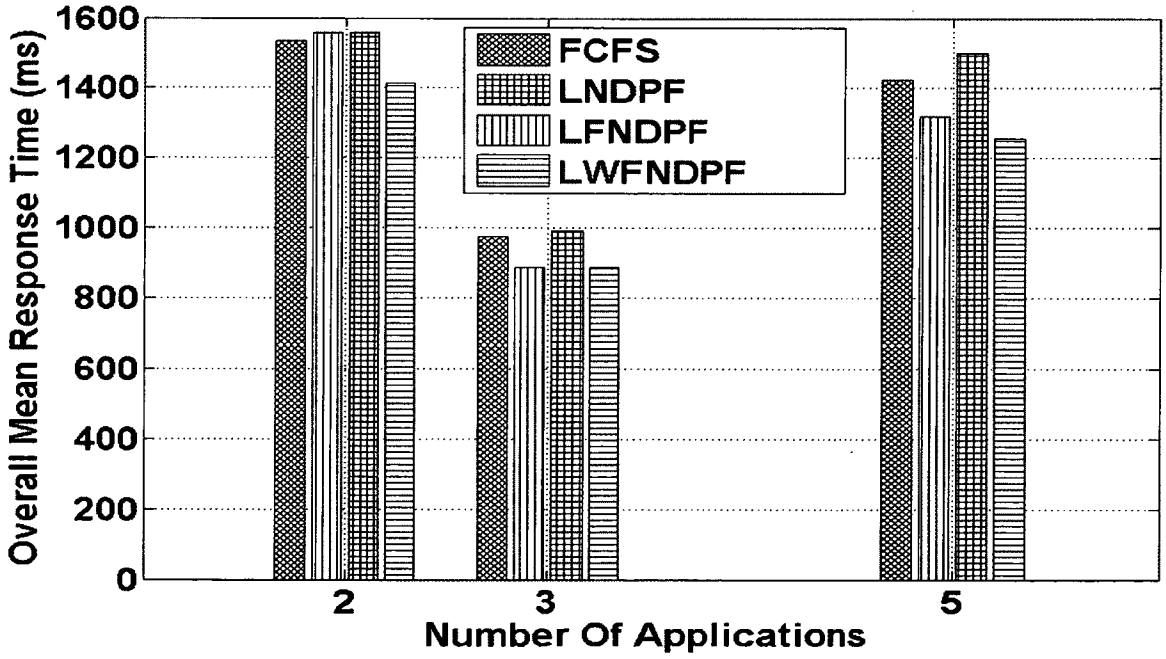


Figure 4-4: Effect of Number of Applications on the Performance of Algorithms

For all the three cases, corresponding to 2, 3, and 5 applications respectively, the LWFNDPF algorithm provides the best performance.

4.4 Effect of Allocation on the Performance of Algorithms

In the default topology, a planned deployment of sensors is assumed. A fixed allocation of sensors at the various hop distances to applications is considered. For the default number of total number of sensors required by the applications (given in Table 4-1), in this experiment, the allocation of sensors to applications at various hop distances is varied. The allocations experimented with are shown in Table 4-5. The default allocation is referred as Allocation 1 in Table 4-5 and corresponds to the allocation of sensor nodes to applications in the default configuration presented in Table 4-3.

Table 4-5: Allocation Details for Scenarios with Planned Deployment

Apps	Allocation 1	Allocation 2	Allocation 3	Allocation 4	Allocation 5
App 1	10, 0, 0, 5	0, 10, 0, 5	0, 0, 10, 5	0, 0, 5, 10	0, 5, 10, 0
App 2	10, 0, 0, 10	0, 10, 0, 10	0, 0, 10, 10	0, 0, 10, 10	0, 10, 10, 0
App 3	10 , 0, 15 , 0	0, 10, 15, 0	0, 15, 10, 0	0, 15, 0, 10	10, 5, 5, 5
App 4	10, 15, 5 , 0	10, 15, 5, 0	10, 5, 15, 0	10, 5, 0, 15	10, 5, 10, 5
App 5	25, 10, 5, 0	15, 20, 5, 0	15, 5, 20, 0	15, 5, 0, 20	5, 15, 10, 10

Each column indicates the number of sensors required at hops 1, 2, 3, and 4 for all the 5 applications for a given allocation. Sensors required at each hop are separated by commas. For example, for Allocation 2 for App 1 "0, 10, 0, 5" means, Application 1 requires no sensor at a distance of 1 hop from the cluster head, 10 sensors at a distance of 2 hops from the cluster head, no sensor at a distance of 3 hops from the cluster head and 5

sensors at a distance of 4 hops from the cluster head. For the five allocations, the number of sensor nodes shared amongst the applications is set to 10. In the default configuration that corresponds to Allocation 1 in Table 4-5, at a distance of 1 hop from the cluster head, each application requires 10 or more sensor nodes. Ten sensor nodes out of the available 25 sensor nodes at a distance of 1 hop from the cluster head are shared amongst the applications. Similar to Allocation 1, for Allocation 2, Allocation 3, Allocation 4, and Allocation 5, the number of sensor nodes shared amongst the five applications is fixed at 10. In Allocation 2, Allocation 3, and Allocation 4, the sensor nodes shared amongst the applications are at a distance of 2 hops from the cluster head, 3 hops from the cluster head, and 4 hops from the cluster head respectively. In Allocation 5, the sensor nodes shared amongst the applications are at a distance of 3 hops or 4 hops from the cluster head.

The performances of the algorithms are shown in Figure 4-5. It may be noted that the results for the various allocations are at different arrival rates as the arrival rate at which the system saturates is different for the various allocations. The mean response times for Allocation 3, 4, and 5 are considerably higher than the mean response times for Allocation 1 and 2 due to the higher number of sensors required by the various applications at a distance of 3 hops and 4 hops from the cluster head. For the allocations experimented with, the LWFNDPF algorithm provides the best performance for 3 allocations, and for the other two allocations, its performance is similar to the best performer.

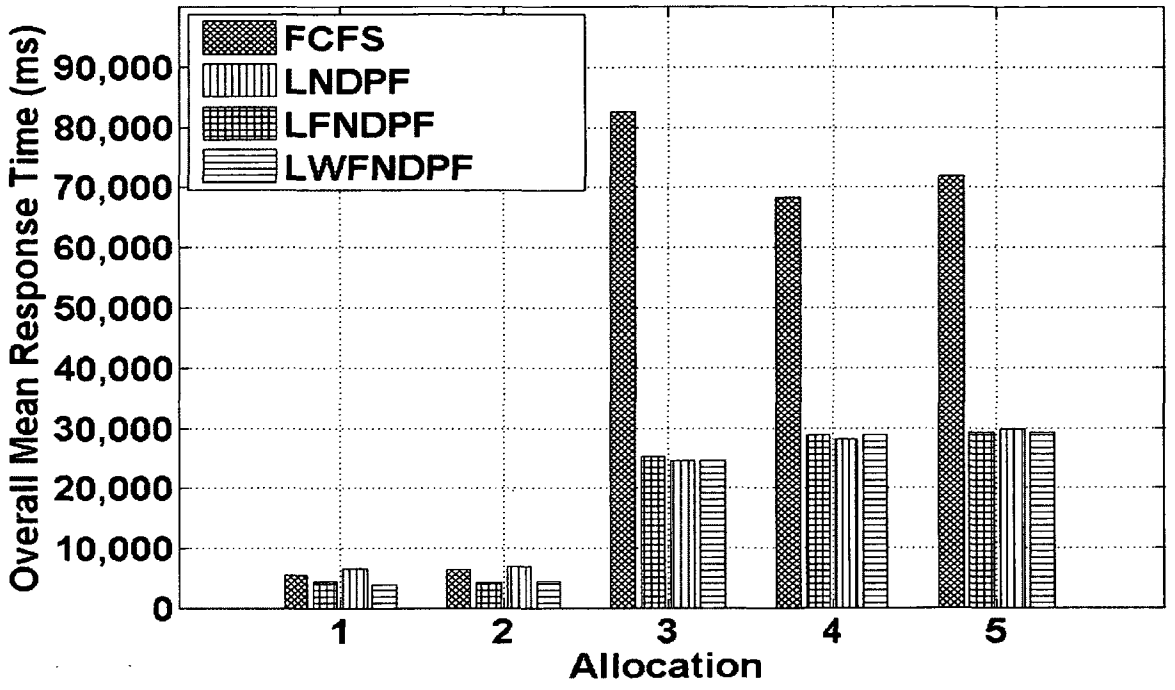


Figure 4-5: Effect of Allocation on the Performance of Algorithms

4.5 Effect of Deployment on the Performance of Algorithms

During the system design phase, the designer has knowledge about the total number of sensor nodes to be deployed in a given area. The deployment of the sensors in the given field may either be random or deterministic/planned [95] [107]. The experiments conducted so far have assumed a planned deployment of sensor nodes in the area to be monitored. Deployment of sensors in office buildings, computer centres, manufacturing plants etc. are usually planned and the exact location of each of the sensors is known. However, for various applications such as forest fire detection it may not be possible to deploy the sensors in a planned manner and the deployment of sensors may be random [107]. Uniform random deployment is a very commonly used deployment which results from throwing sensor nodes from an aeroplane [107]. With such a deployment each of the sensors has equal probability of being placed at any point

in a given field. As such a deployment is very common in wireless sensor networks, simulation experiments have been performed to study the performance of the scheduling algorithms proposed in this research for uniform random deployment of sensor nodes in a given field. Eight configurations have been generated corresponding to uniform random deployment. The number of applications and the number of sensors required by each application are held at the default values. The sensor nodes are deployed in a given area with their x and y coordinates following a uniform distribution. After the sensors have been deployed, the sensors that will serve the various applications are selected at random based on the total number of sensors required by each application. Table 4-6 gives the details of allocation of sensors at each hop for the various applications for each configuration. In the first column, number of sensors at each hop distance is mentioned in parentheses separated by commas. Column 2, Column 3, Column 4, Column 5, and Column 6 give the number of sensors required by each application within parentheses. The values separated by commas denote the number of sensors required by the respective applications at a distance of 1 hop, 2 hops, 3 hops, and 4 hops from the cluster head respectively. Based on the allocation of the sensor nodes to various applications at each hop distance, the Number Distance Product, the Farthest Number Distance Product, and the Weighted Farthest Number Distance Product for each application is computed. The computed metrics for the applications are used by the various scheduling algorithms to schedule requests for the various applications. It may be recalled that NDP is the product of the total number of sensors required by the application and the average distance of the required sensors from the cluster head. The Farthest Number Distance Product is the product of the number of sensors required by the application that are located farthest

away from the cluster head and the farthest distance in terms of number of hops from the cluster head. The WFNDP is the product of FNDP for the application and the respective weight at the farthest hop distance.

Table 4-6: Allocation Details for Scenarios with Random Deployment

Configuration	Application 1	Application 2	Application 3	Application 4	Application 5
Configuration 1 (27, 24, 27, 22)	(2, 6, 3, 4)	(4, 3, 7, 6)	(9, 4, 9, 3)	(9, 4, 10, 7)	(10, 8, 12, 10)
Configuration 2 (36, 22, 22, 20)	(6, 4, 3, 2)	(11, 3, 3, 3)	(7, 6, 6, 6)	(12, 6, 6, 6)	(16, 9, 7, 8)
Configuration 3 (30, 23, 27, 20)	(3, 4, 6, 2)	(7, 7, 4, 2)	(7, 4, 6, 8)	(9, 5, 6, 10)	(11, 15, 10, 4)
Configuration 4 (35, 21, 27, 17)	(7, 5, 2, 1)	(6, 5, 7, 2)	(8, 5, 8, 4)	(11, 6, 8, 5)	(12, 11, 7, 10)
Configuration 5 (22, 31, 24, 23)	(3, 6, 1, 5)	(6, 4, 3, 7)	(3, 13, 5, 4)	(5, 11, 7, 7)	(7, 10, 13, 10)
Configuration 6 (42, 26, 24, 8)	(3, 9, 3, 0)	(9, 6, 3, 2)	(11, 6, 6, 2)	(13, 7, 7, 3)	(16, 8, 11, 5)
Configuration 7 (30, 52, 18, 0)	(3, 11, 1, 0)	(6, 10, 4, 0)	(8, 14, 3, 0)	(7, 18, 5, 0)	(10, 20, 10, 0)
Configuration 8 (10, 52, 18, 20)	(0, 4, 6, 5)	(0, 4, 9, 7)	(1, 6, 13, 5)	(3, 6, 17, 3)	(0, 10, 23, 7)

The performance of the proposed scheduling algorithms for the eight configurations experimented with is shown in Figure 4-6 to Figure 4-13.

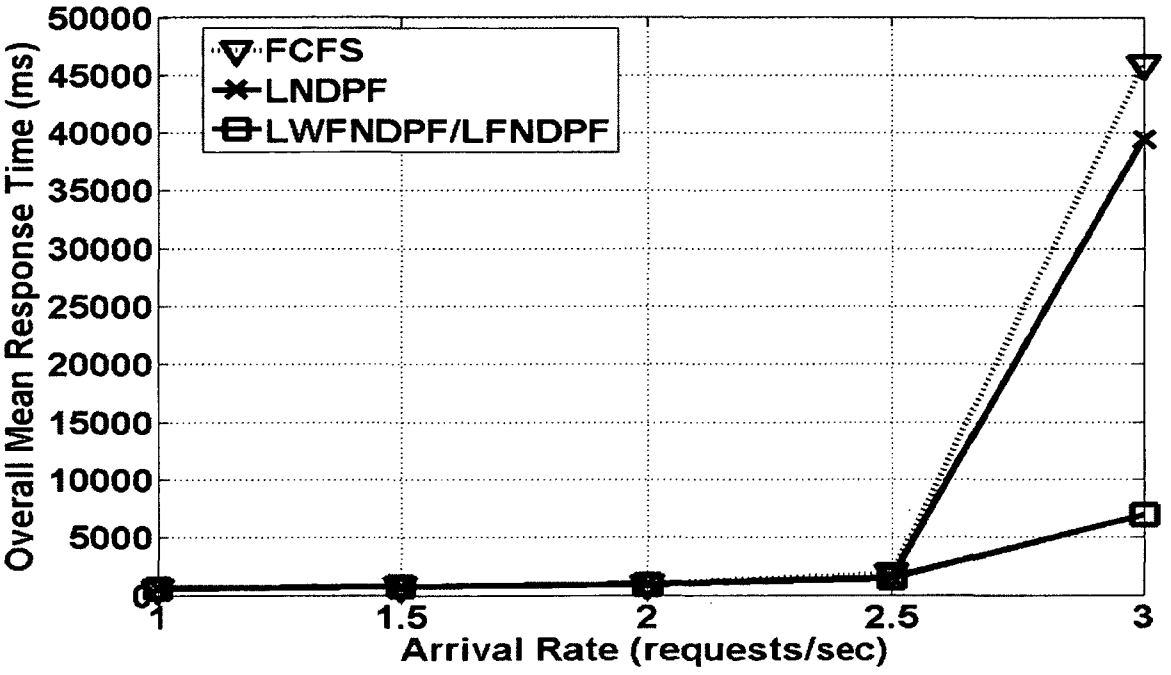


Figure 4-6: Performance of Algorithms in Configuration 1

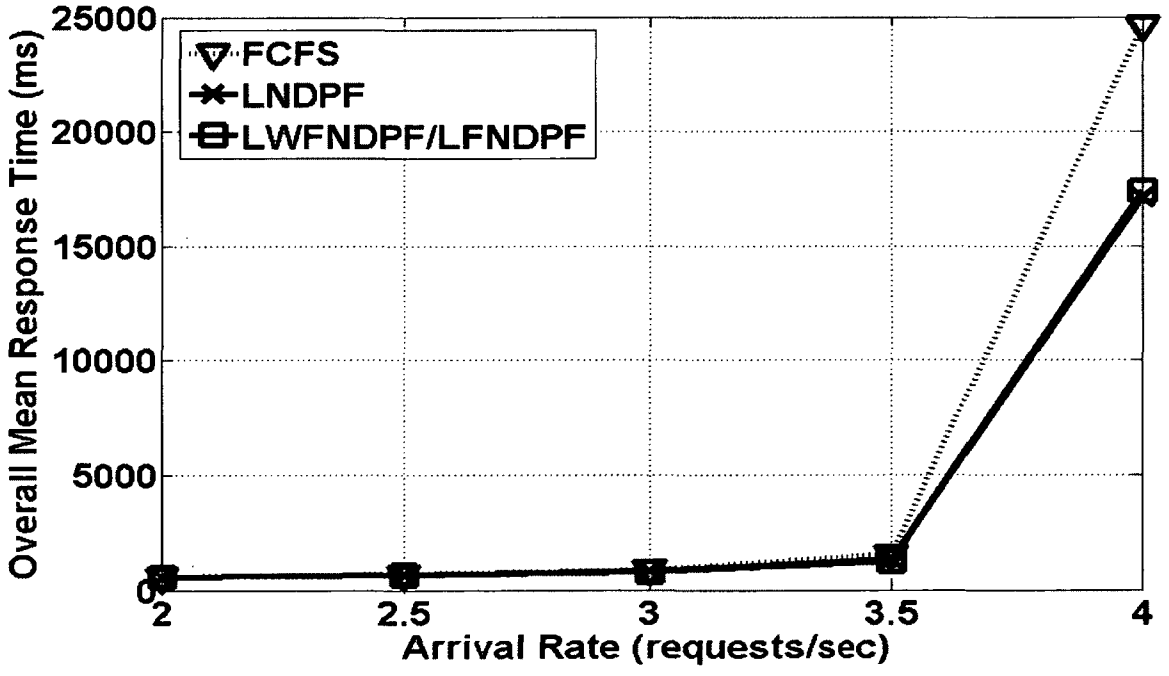


Figure 4-7: Performance of Algorithms in Configuration 2

LWFNDPF provides the best performance for configurations 1, 2, 3, 4, 5, and 8.

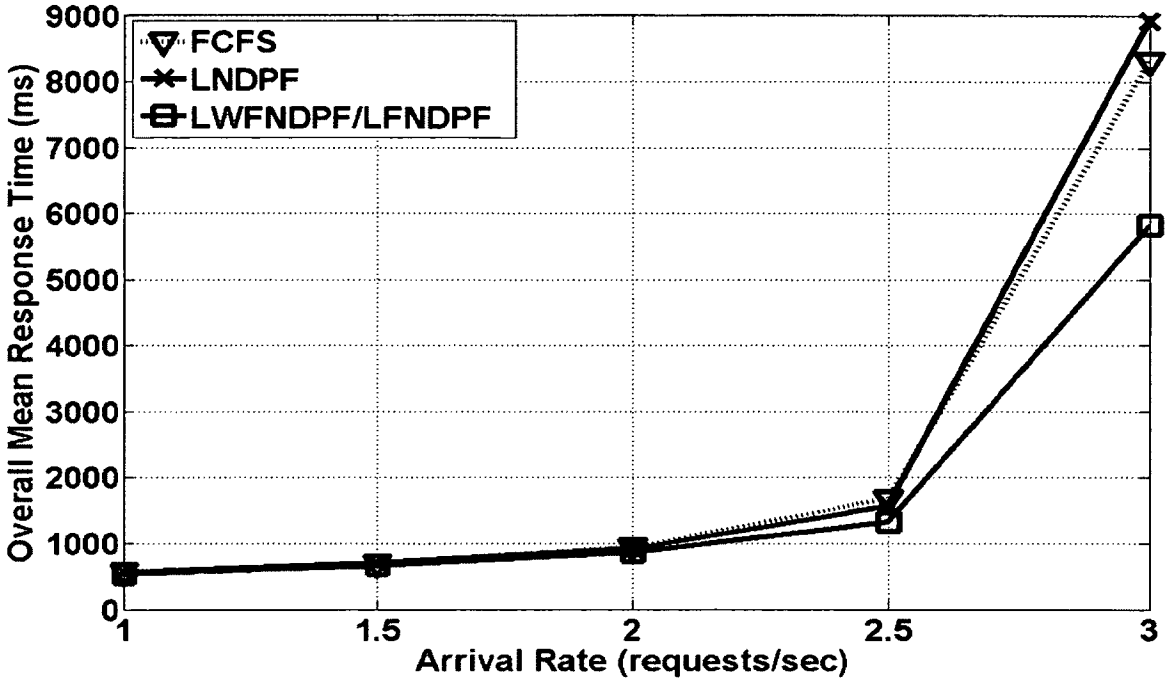


Figure 4-8: Performance of Algorithms in Configuration 3

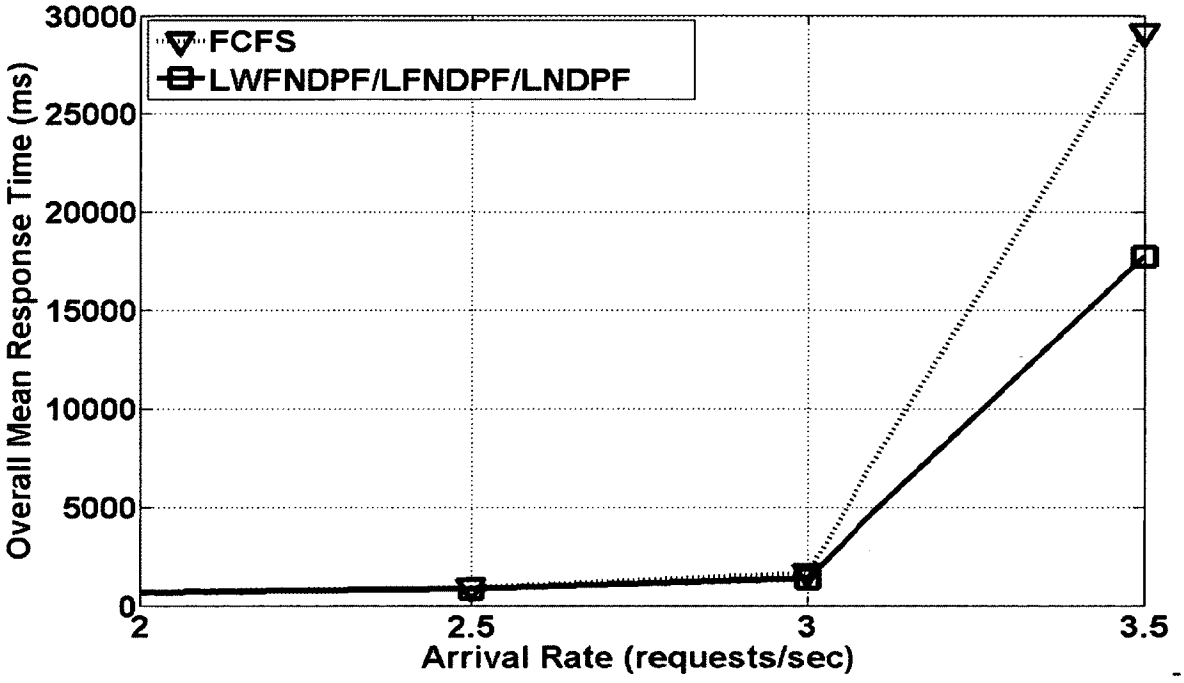


Figure 4-9: Performance of Algorithms in Configuration 4

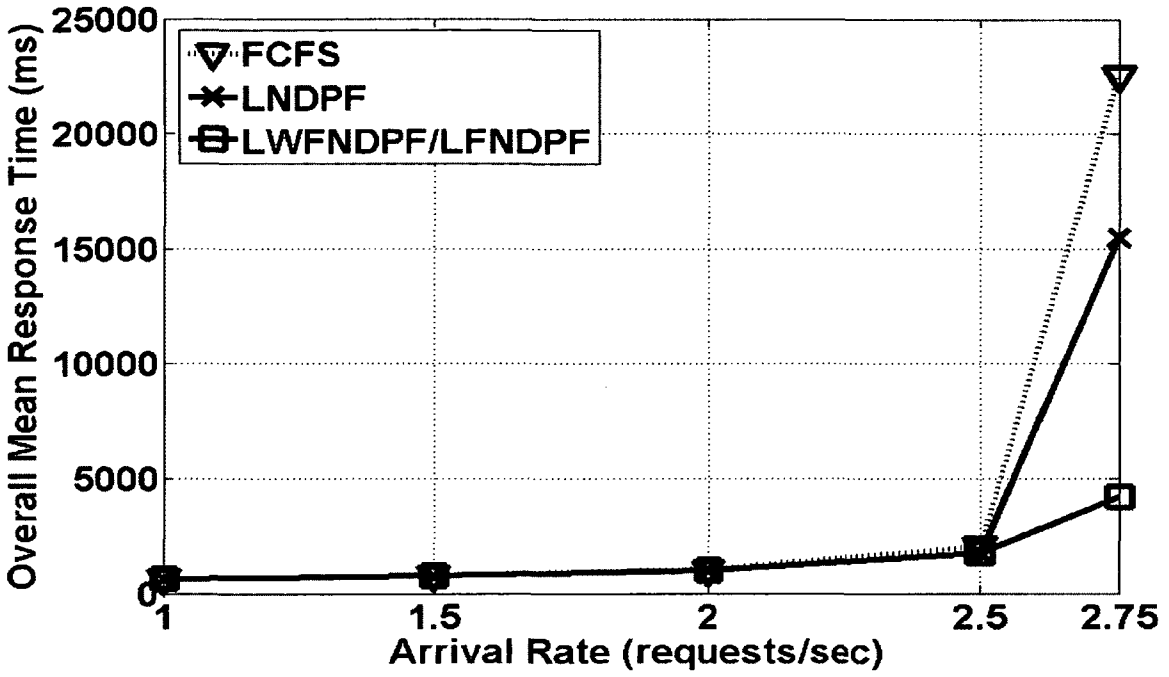


Figure 4-10: Performance of Algorithms in Configuration 5

For configuration 6 and configuration 7, performance of the LWFNDPF algorithm is very close to the best performer.

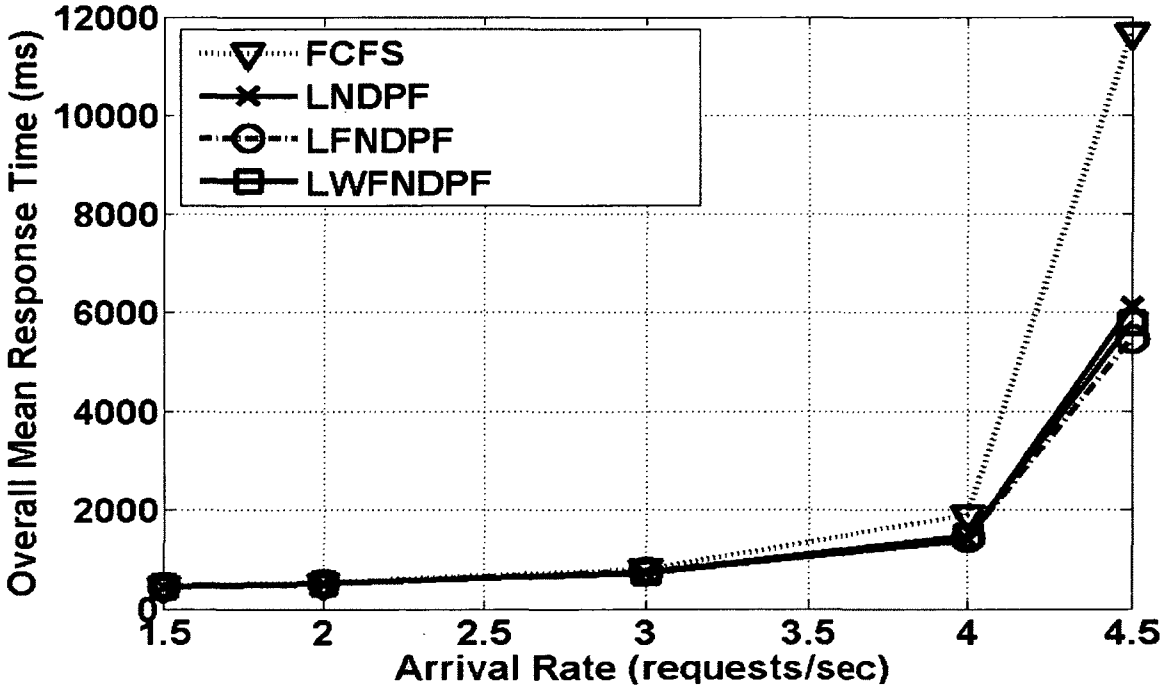


Figure 4-11: Performance of Algorithms in Configuration 6

For configuration 6, the overall mean response time provided by LWFNDPF is only 5.5% higher than the overall mean response time provided by the best performer.

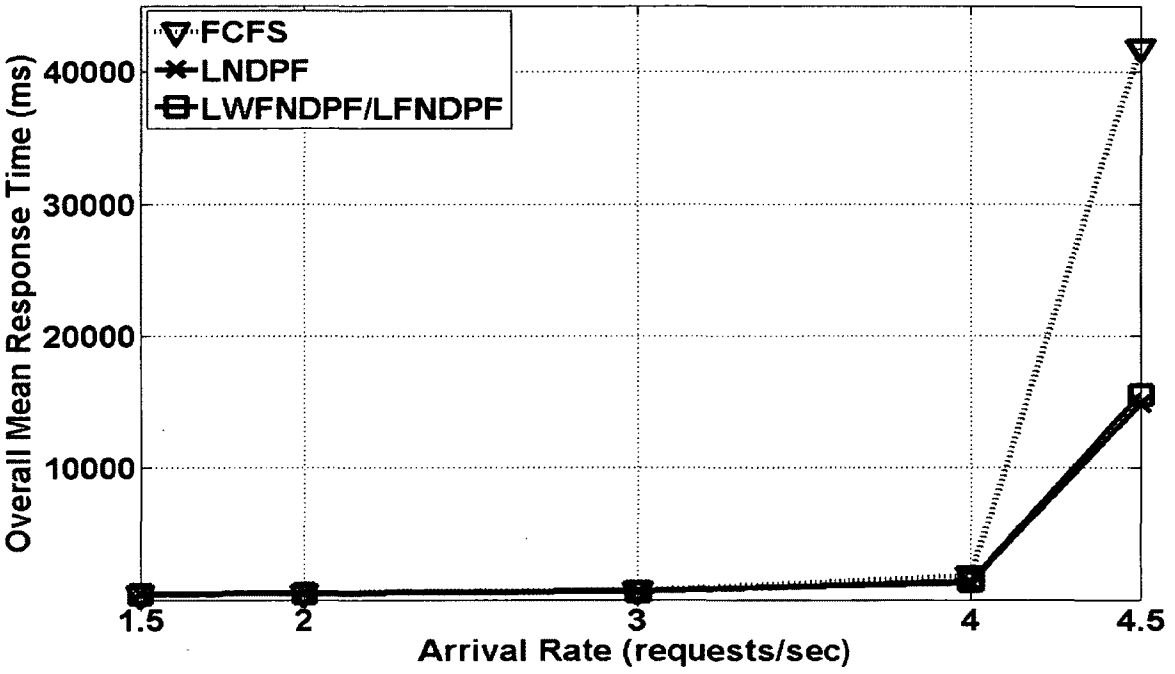


Figure 4-12: Performance of Algorithms in Configuration 7

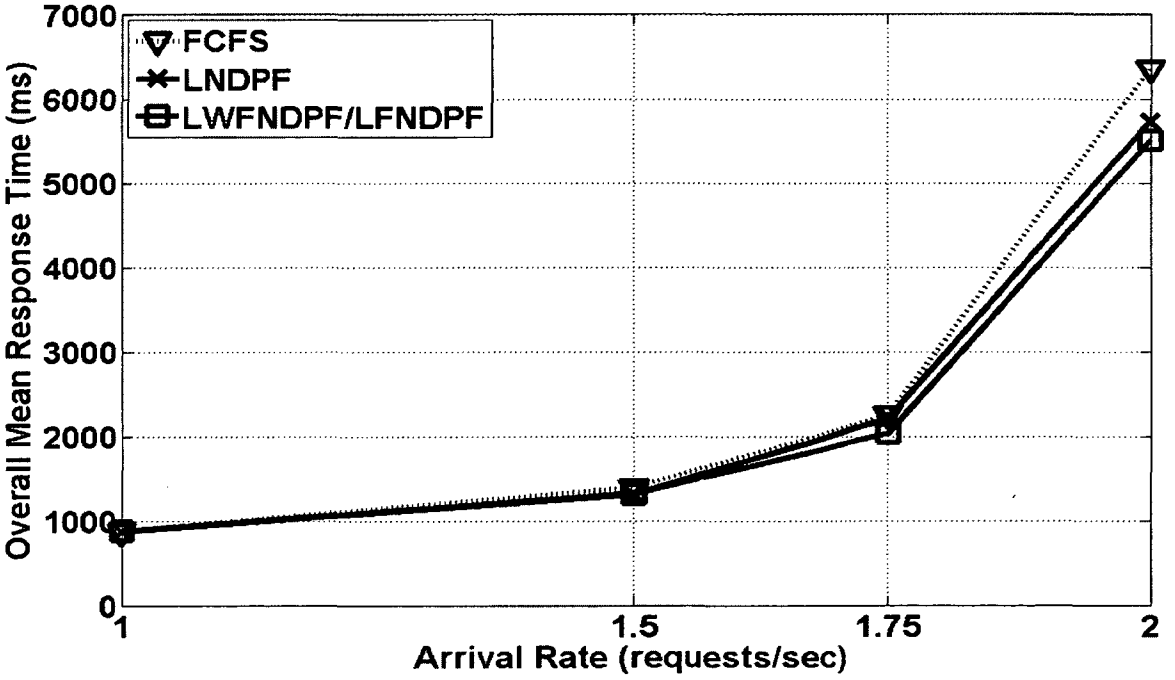


Figure 4-13: Performance of Algorithms in Configuration 8

For configuration 7, the overall mean response time provided by the LWFNDPF algorithm is only 4.5% higher than the overall mean response time provided by the best performer.

4.6 Effect of Unequal Arrival Rates from Various Applications

In the default configuration, requests from all the five applications have an equal probability of arrival. The probability of arrival of request from each of the applications is set to 0.2. Experiments have been performed to study the effect of unequal arrival rates of requests for various applications on the performance of the proposed algorithms. Two scenarios have been considered. In Scenario 1, it has been assumed that requests for applications requiring a lower number of sensors have a higher probability of arrival and in Scenario 2, requests for applications requiring a higher number of sensors have a higher probability of arrival. In Scenario 1, the probabilities of arrival of request for Application 1, Application 2, Application 3, Application 4, and Application 5 are set to 0.4, 0.25, 0.2, 0.1, and 0.05 respectively. The relative performances of the various scheduling algorithms are shown in Figure 4-14. The LWFNDPF algorithm provides the best performance. In Scenario 2, the probabilities of arrival of request for Application 1, Application 2, Application 3, Application 4, and Application 5 are set to 0.05, 0.1, 0.2, 0.25, and 0.4 respectively. The relative performances of the various scheduling algorithms in Scenario 2 are shown in Figure 4-15.

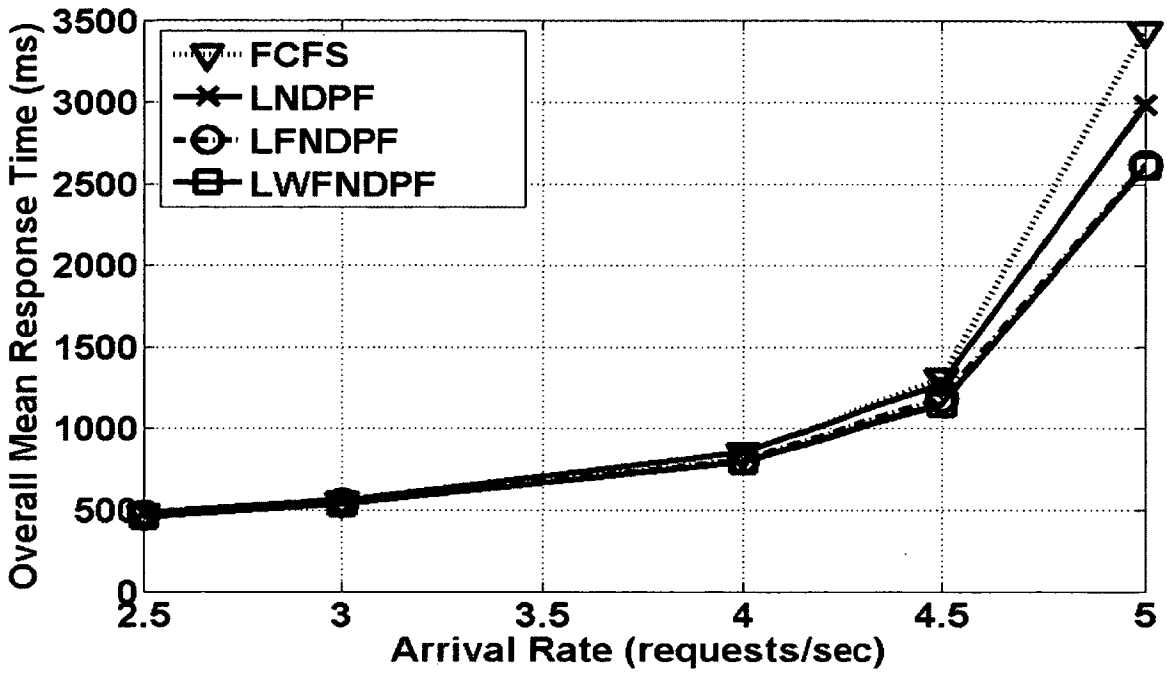


Figure 4-14: Effect of Unequal Arrival Rates from Applications in Scenario 1

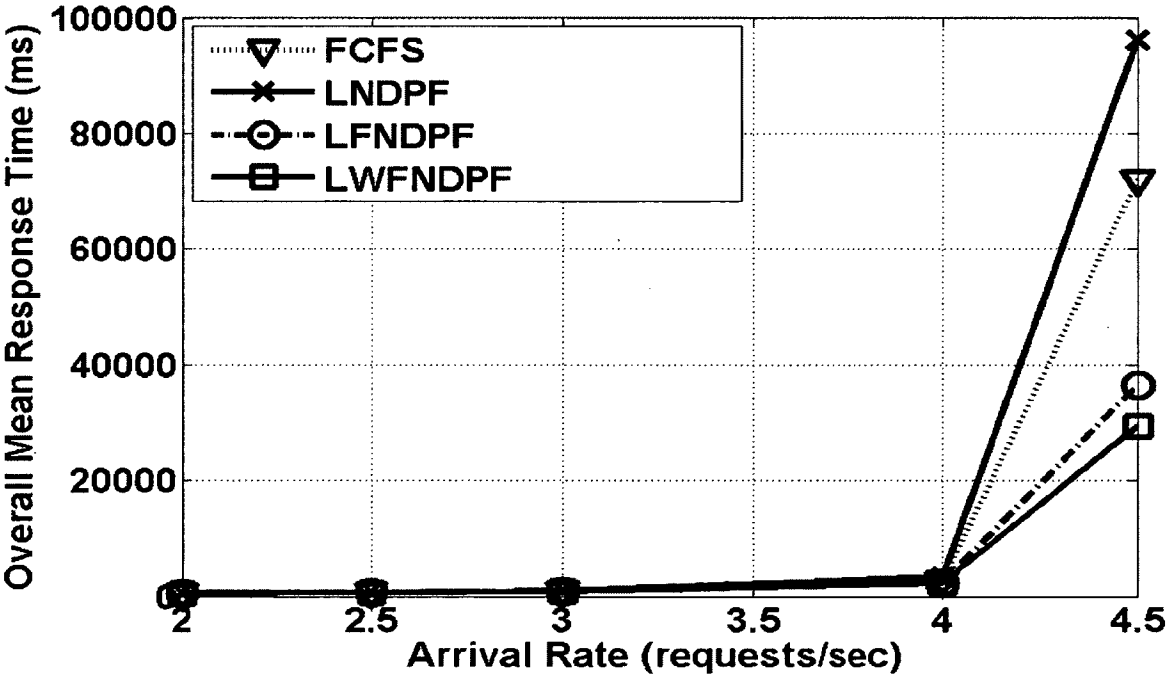


Figure 4-15: Effect of Unequal Arrival Rates from Applications in Scenario 2

The LWFNDPF algorithm provides the best performance in Scenario 2 also. The LNDPF algorithm demonstrates the worst performance. The difference in the performances of the algorithms for Scenario 2 is greater as compared to Scenario 1. This

may be attributed to the fact that the LWFNDPF algorithm favors requests from Application 4 and Application 5 as these applications have the smallest value of the Weighted Farthest Number Distance product. In Scenario 2, requests from Application 4 and Application 5 also have a higher probability of arrival and hence the LWFNDPF algorithm that favors requests from these applications performs much better than the other algorithms. The LNDPF algorithm favors requests from applications that have lesser probability of arrival and hence provides the worst performance (see Figure 4-15).

4.7 Effect of MAC Layer Protocol on the Performance of Algorithms

Most of the experimentation done in this thesis has been done with the assumption of IEEE 802.11 as the underlying MAC layer protocol. In order to study the effect of the underlying MAC layer protocol on the performance of the scheduling algorithms proposed in this thesis, an abstraction of the SMAC protocol has been implemented. SMAC [55] is a contention based protocol based on the IEEE 802.11 protocol. In order to conserve energy of the resource constrained sensor nodes, the SMAC protocol lets the nodes go into a periodic sleep mode. The sensor nodes alternate between sleep cycles and listen cycles. In the experiment, the durations of the sleep interval and the listen interval are kept equal and are fixed at 0.3 seconds. Similar values of sleep interval and listen interval have been used by other researchers [55].

The performance of the proposed algorithms in the default configuration with SMAC as the underlying protocol is shown in Figure 4-16. The SMAC protocol adds an additional latency due to the periodic sleep cycles of the sensor nodes as the sensor nodes do not transmit or receive messages during the sleep cycles. The overall mean response

time for all the algorithms with SMAC as the underlying protocol is higher than the overall mean response time obtained with IEEE 802.11 as the underlying protocol (see Figure 4-1 and Figure 4-16). Also, for a given arrival rate, the difference in the performance of the various algorithms is greater with SMAC as the underlying protocol as compared to the difference in the performance of the algorithms with IEEE 802.11 as the underlying protocol. Similar to the IEEE 802.11 protocol, the LWFNDPF algorithm demonstrates the best performance even with SMAC as the underlying MAC protocol.

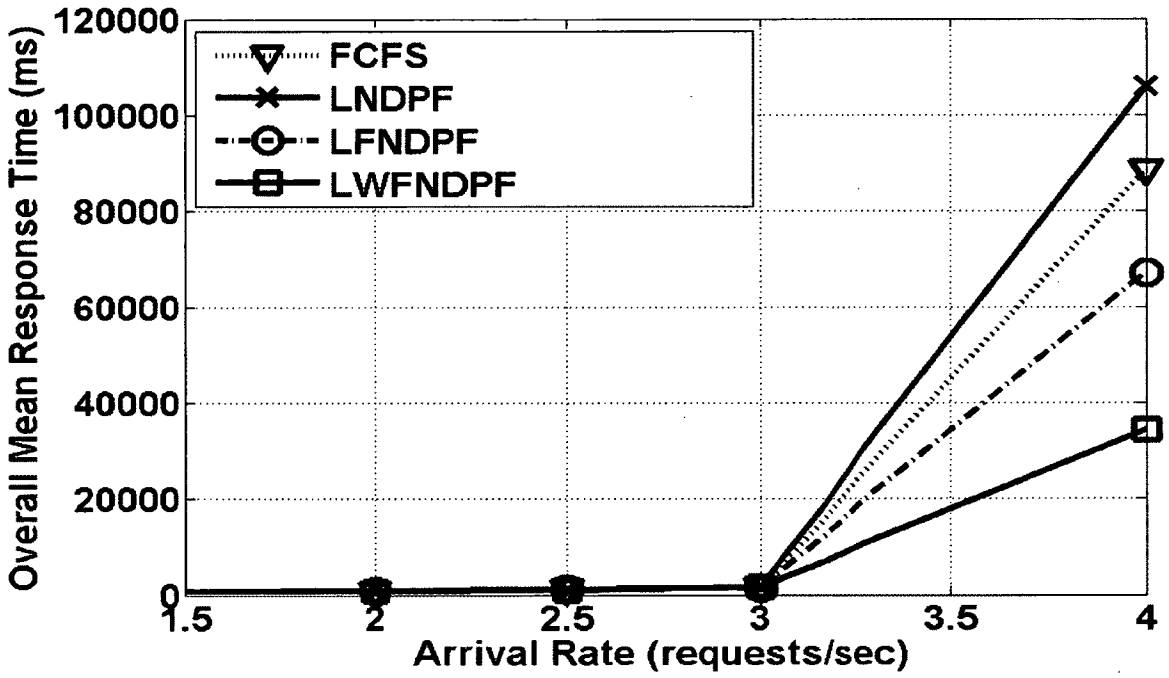


Figure 4-16: Effect of Arrival Rate on the Performance of Algorithms with SMAC as the MAC Layer Protocol

The default configuration corresponds to a planned deployment of sensors in the sensor field. An experiment was also conducted for a scenario corresponding to uniform random deployment of sensors in the sensor field with SMAC as the underlying MAC layer protocol. This experiment corresponds to Configuration 1 presented in Section 4-5.

Comparison of performances of the algorithms with the IEEE 802.11 protocol and the SMAC protocol as the underlying MAC layer protocol is presented in Figure 4-17.

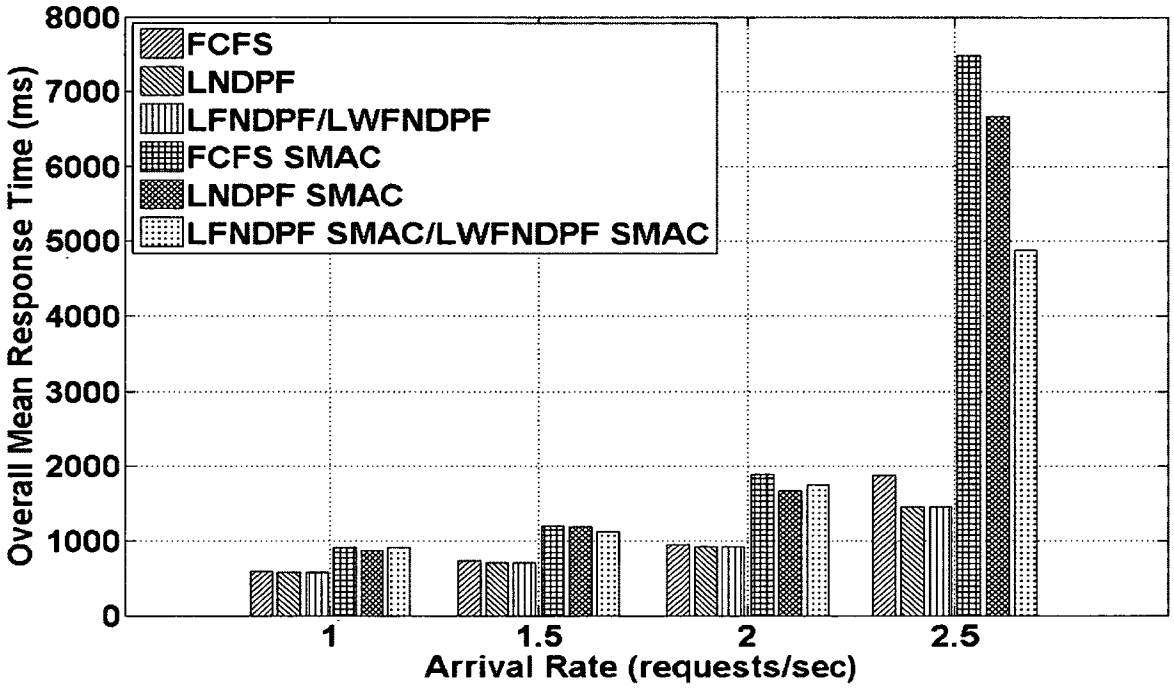


Figure 4-17: Effect of MAC Layer Protocol on the Performance of Algorithms in a Random Deployment Scenario

It may be noted that the performance of the LWFNDPF algorithm and the LFNDPF algorithm are identical. In this configuration, all the applications have the farthest sensors at a distance of 4 hops from the cluster head. The allocated weight at a distance of 4 hops from the cluster head is 1. Therefore, all the applications have identical values of FNDP and WFNDP which leads to the same performances of the LWFNDPF algorithm and the LFNDPF algorithm. Similar to the earlier experiment corresponding to planned deployment, the overall mean response time for all algorithms increases with SMAC as the underlying protocol. This can be attributed to the additional latency added by the periodic sleep cycles in the SMAC protocol. Also, the difference in the performances of the algorithms with IEEE 802.11 and SMAC as the underlying MAC

protocol increases with the increase in the arrival rate. This can be attributed to the increase in the number of messages waiting to be transmitted with the increase in the arrival rate of application requests due to the sleep cycle in SMAC.

4.8 Performance Analysis of Scheduling Algorithms

A schedule is an order in which the job requests allocated to a particular sensor node can be executed. How well do the proposed algorithms perform in comparison to the best system performance that can be achieved is an important question. This section analyses this question by performing an exhaustive search in the entire solution space that includes all possible schedules. To achieve a reasonable time for performing this experiment, a system deploying three applications (Application 1, Application 2 and Application 4) is considered first and then a system deploying four applications (Application 2, Application 3, Application 4, and Application 5) is considered. For the three application case that considers a system deploying Application 1, Application 2, and Application 4, based on the default topology, Application 4 has the highest NDP but the lowest FNDP from amongst the three applications. This configuration is particularly selected to validate the observation that the sensors placed farthest away from the cluster head play the most important role in determining the overall mean response time. The arrival rate is equally split amongst the applications considered. For the three applications case, the six possible schedules for the applications are: Order 1 (1, 2, 4), Order 2 (1, 4, 2), Order 3 (2, 1, 4), Order 4 (2, 4, 1), Order 5 (4, 1, 2), and Order 6 (4, 2, 1) where the numbers i = 1-5 inside the brackets specify the order in which the requests for Application i are scheduled. Order 5 (4, 1, 2) also corresponds to the order in which the

requests would be scheduled if the Least Weighted Farthest Number Distance Product First scheduling algorithm is deployed. The overall mean response time of the applications for the six schedules is shown in Figure 4-18. Based on the default deployment, Application 4 has the smallest value of WFNDP and Application 2 has the largest value of WFNDP amongst the three applications that are considered here.

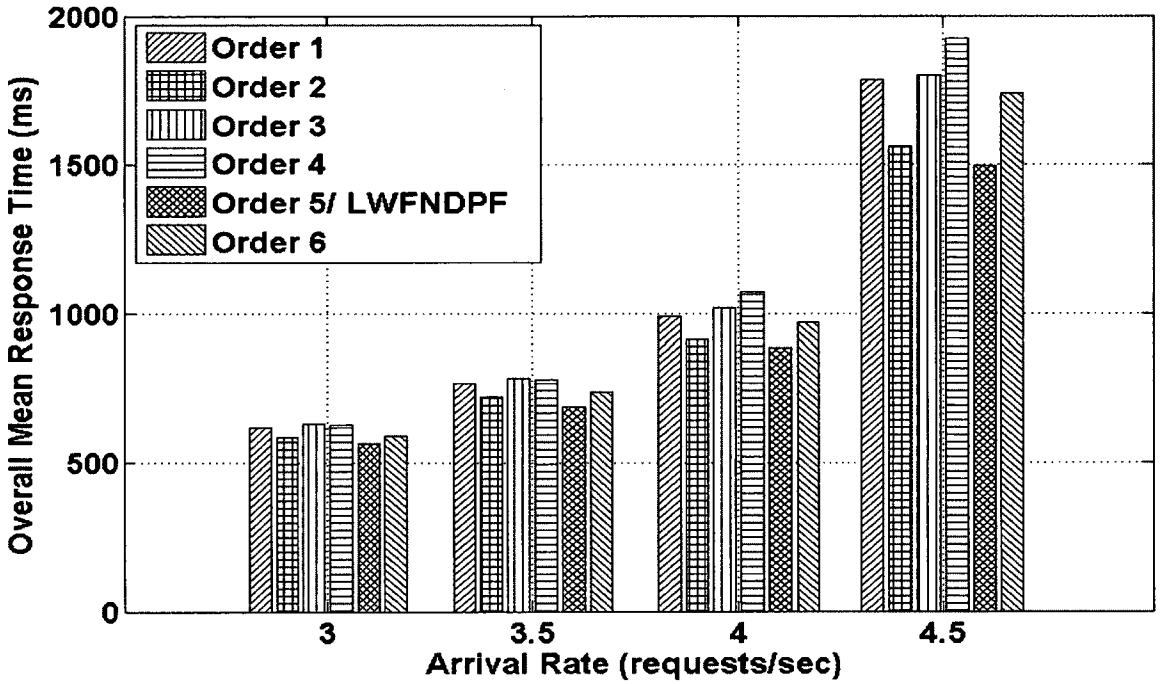


Figure 4-18: Performance of Algorithms for a Three Application System

It is encouraging to note that Order 5 (4, 1, 2), which is the order that corresponds to the LWFNDPF algorithm provides the best performance.

For the four application case, Application 2, Application 3, Application 4, and Application 5 are considered. Based on the default topology (see Table 4-3), Application 3, and Application 4 have the same value of NDP. However, Application 3 has a higher value of FNDP and WFNDP as compared to Application 4. Application 4 and Application 5 have same value of FNDP and WFNDP but unequal values for NDP. This set of applications is useful in determining that the sensors located farther away play a

significant role in determining the system performance. Once again, for the four applications case, all the possible schedules are considered. For the four application system deploying Application 2, Application 3, Application 4, and Application 5, the possible schedules for the applications are: Order 1 (2, 3, 4, 5), Order 2 (2, 3, 5, 4), Order 3 (2, 4, 3, 5), Order 4 (2, 4, 5, 3), Order 5 (2, 5, 3, 4), Order 6 (2, 5, 4, 3), Order 7 (3, 2, 4, 5), Order 8 (3, 2, 5, 4), Order 9 (3, 4, 2, 5), Order 10 (3, 4, 5, 2), Order 11 (3, 5, 2, 4), Order 12 (3, 5, 4, 2), Order 13 (4, 2, 3, 5), Order 14 (4, 2, 5, 3), Order 15 (4, 3, 2, 5), Order 16 (4, 3, 5, 2), Order 17 (4, 5, 2, 3), Order 18 (4, 5, 3, 2), Order 19 (5, 2, 3, 4), Order 20 (5, 2, 4, 3), Order 21 (5, 3, 2, 4), Order 22 (5, 3, 4, 2), Order 23 (5, 4, 2, 3), and Order 24 (5, 4, 3, 2). For the LNDPF algorithm, LFNDPF algorithm, and the LWFNDPF algorithm, the tie is broken using the First Come First Served paradigm for applications having the same value of Number Distance Product, Farthest Number Distance Product, and the Weighted Farthest Number Distance Product respectively. Amongst the applications considered in the four application case, Application 3 and Application 4 have the same value of NDP. Also, Application 4 and Application 5 have the same value of FNDP and WFNDP. The overall mean response times obtained with the twenty-four possible schedules are given in Table 4-7 for three arrival rates. The overall mean response times obtained with the LNDPF algorithm, the LFNDPF algorithm, the LWFNDPF algorithm, and the FCFS algorithm are given in Table 4-8 for three arrival rates. The results are presented in a tabular form as presenting 28 bars for three arrival rates on a bar graph does not provide enough clarity.

Table 4-7: Overall Mean Response Time for Various Schedules in Four Application Case

Scheduling Order	Arrival Rate = 3.5 requests/sec	Arrival Rate = 4 requests/sec	Arrival Rate = 4.5 requests/sec
Order 1 (2, 3, 4, 5)	1242.023	2818.697	81778.240
Order 2 (2, 3, 5, 4)	1224.777	2525.932	61567.451
Order 3 (2, 4, 3, 5)	1235.376	2668.442	80507.372
Order 4 (2, 4, 5, 3)	1206.601	2407.121	57569.07985
Order 5 (2, 5, 3, 4)	1216.697	2569.841	47316.939
Order 6 (2, 5, 4, 3)	1220.227	2371.195	55721.54359
Order 7 (3, 2, 4, 5)	1186.971	2773.313	83832.74213
Order 8 (3, 2, 5, 4)	1198.67	2448.718	61385.15325
Order 9 (3, 4, 2, 5)	1161.804	2572.117	75035.6304
Order 10 (3, 4, 5, 2)	1172.093	1855.317	50272.89208
Order 11 (3, 5, 2, 4)	1169.624	2220.354	46721.22463
Order 12 (3, 5, 4, 2)	1180.0224	2048.431	48115.817
Order 13 (4, 2, 3, 5)	1147.949	2606.633	78540.688
Order 14 (4, 2, 5, 3)	1128.054	2195.682	50541.47
Order 15 (4, 3, 2, 5)	1143.93	2567.629	79190.195
Order 16 (4, 3, 5, 2)	1143.115	1859.845	57945.973
Order 17 (4, 5, 2, 3)	1115.148	2012.389	38186.525
Order 18 (4, 5, 3, 2)	1143.165	2008.684	47465
Order 19 (5, 2, 3, 4)	1037.221	2303.867	38160.065
Order 20 (5, 2, 4, 3)	1055.121	2218.304	40692.018
Order 21 (5, 3, 2, 4)	1040.88	2300.809	38031.278
Order 22 (5, 3, 4, 2)	1078.194	2077.836	41889.778
Order 23 (5, 4, 2, 3)	1071.446	2137.977	38159.407
Order 24 (5, 4, 3, 2)	1117.949	2111.982	42026.926

Table 4-8: Overall Mean Response Time for Various Algorithms in Four Application Case

Algorithm	Arrival Rate = 3.5 requests/sec	Arrival Rate = 4 requests/sec	Arrival Rate = 4.5 requests/sec
LNDPF	1221.081	2836.733	80133.32937
LFNDPF	1123.836	2124.906	46243.713
LWFNDPF	1095.944247	2017.18	41724.11452
FCFS	1179.473	2448.685	74897.39556

At an arrival rate of 3.5 requests/second, only five orders provide a better overall mean response time than the LWFNDPF algorithm. However, the difference between the overall mean response time provided by the LWFNDPF algorithm and the best overall mean response time obtained is only 5.2%.

At an arrival rate of 4 requests/second, only four schedules provide a better overall mean response than the LWFNDPF algorithm. Improvement provided by two schedules is within 1% of the overall mean response time provided by the LWFNDPF algorithm. Improvement in the overall mean response time provided by the other two schedules as compared to the LWFNDPF algorithm is around 8%.

At an arrival rate of 4.5 requests/second, five schedules provide a better performance than the LWFNDPF algorithm. Four schedules provide an improvement close to 8.5 % in response time whereas for one schedule the improvement is close to only 2.5%. The performance of the LWFNDPF algorithm is close to the best performer.

4.9 Discussion of Simulation Results

In this section, a brief discussion on the simulation experiments presented earlier in this chapter is provided.

Experiments have been performed by varying the arrival rate of requests for applications, the data size of request and response messages, and the number of applications hosted by the WSN. Effect of allocation on the performance of algorithms has also been studied. Experiments have been performed to study the performance of the proposed scheduling algorithms, both in case of a planned deployment of sensor nodes and also in case of a random deployment of sensor nodes in the area to be monitored. Effect of the underlying MAC layer protocol on the performance of the algorithms has been studied. Performance of the scheduling algorithms is compared with the best achievable performance determined through an exhaustive search in the entire solution space that includes all possible schedules.

The important insights generated from the experiments performed are now summarized.

- For most of the configurations experimented with, the LWFNDPF algorithm demonstrates the best performance.
- The difference in the performance of the algorithms increases with the increase in the arrival rate of requests for applications.
- For a given arrival rate, as the data size of the request and response messages is increased, the overall mean response time increases for all the algorithms. The difference in the performances of the algorithms increases with the increase in the data size of the request and response messages. This is because

the transmission time of the messages increases with the increase in the size of messages, the messages waiting to be transmitted experience greater delays at each hop.

- The number of applications hosted by the WSN is varied from 2 to 3 to 5. Irrespective of the number of applications hosted by the WSN, the LWFNDPF algorithm demonstrated the best performance.
- For a planned deployment, various allocations are experimented with. Keeping the total number of sensors required by the applications fixed at the default values (see Table 4-1); the number of sensors required by the applications at the various hop distances from the cluster head is varied. For the allocations experimented with, the LWFNDPF algorithm either provides the best performance or its performance is comparable to the best performance.
- Experiments have been performed for various configurations in which the sensor nodes are deployed at random in the area to be monitored. Once again, the LWFNDPF algorithm demonstrates the best performance or its performance is close to the best performance.
- Effect of the underlying MAC layer protocol on the performance of the algorithms is studied. Two configurations, one corresponding to a planned deployment of sensor nodes in the area to be monitored and the other corresponding to a random deployment of sensor nodes were investigated. For both of these configurations, the performances of the algorithms are evaluated for both IEEE 802.11 and an abstraction of the SMAC protocol used as the

underlying MAC layer protocol. The LWFNDPF algorithm demonstrates the best performance irrespective of the underlying MAC layer protocol. However, with the SMAC protocol the overall mean response time is higher for all the algorithms as compared to the overall mean response time with IEEE 802.11. Also, the difference in the performance of the algorithms is observed to be higher with the SMAC protocol.

- Performance of the proposed algorithms is analysed by comparing the performance of the proposed algorithms with all possible schedules for a three application case and a four application case. For the three application case, the scheduling order that corresponds to the LWFNDPF algorithm provides the best performance. For the four application case, at a high arrival rate, the overall mean response time obtained with the LWFNDPF algorithm is the smallest for nineteen schedules out of the total twenty-four possible schedules and at most 8.5% higher than the overall mean response time obtained with the better performing schedule in the other cases.

Chapter 5: Performance of Allocation Algorithms

In this chapter, simulation experiments performed to analyse the performance of the allocation algorithms proposed in this research are presented.

The aim of an allocation algorithm is to increase the lifetime of the WSN. The lifetime of a wireless sensor network can be improved by improving the lifetime of the individual sensors comprising the WSN. The simulation model used in this research has been described Chapter 3. In order to study the effect of a given parameter on performance, a factor at a time approach has been used in our simulation experiments. The parameter of interest is varied while all the other parameters are held at their default levels (see Table 5-1). For the default configuration, the two applications considered have identical characteristics in terms of their resource requirements, execution time, and data size of messages. This has been done to be able to analyse the effect of application and network related parameters on performance separately. In some of the experiments discussed in this chapter, parameters are varied for one application at a time to evaluate the performance of the proposed algorithms for WSNs hosting applications with different characteristics. The energy source in a sensor is a battery. The two components that cause energy drain in a sensor node are the central processing unit and the radio. The initial value of energy of the battery, battery voltage, and the current drawn by the central processing unit and the radio in various states are given in Table 5-2. Similar values have been used by other researchers [108]. The physical and MAC layer parameters take their default values as given in Table 4-2 [88]. The underlying MAC protocol is assumed to be IEEE 802.11. Once a sensor node is allocated to requests for applications, the energy

consumption at a sensor node due to the execution of the requests at the sensor node does not depend upon the MAC protocol and will remain the same irrespective of the underlying MAC protocol.

Table 5-1: Default Values of System and Workload Parameters

Parameter	Value
p	0.5
λ_{appl}, λ_{app2}	$p\lambda$ (0.5λ)
N_s	100
$N_{applications}$	2
S_{appl}	20 (5 sensors at each hop distance)
S_{app2}	20 (5 sensors at each hop distance)
Data Size	100 bytes
E_{appl}, E_{app2}	5 ms

Table 5-2: Default Values of Battery, CPU, and Radio Components

Component	Parameter	Value
Battery	Voltage	3 V
Battery	Initial Energy	1000 Joules
Radio	Transmit Current	100 mA
Radio	Receive Current	8 mA
CPU	Idle Current	3.2 mA
CPU	Active Current	8 mA

The allocation algorithms proposed in this research are summarised in Table 5-3.

Table 5-3: Allocation Algorithms

Allocation	Algorithm	Acronym	Objective	Reference
Static	Random Allocation	RA	Allocate sensor nodes at random.	Section 3.6.1.1
	CPU Load Balanced Allocation	CLBA	Balance the energy consumption at the CPU component amongst the sensor nodes.	Section 3.6.1.2
	Data Load Balanced Allocation	DLBA	Balance the energy consumption at the radio component due to the transmission of response messages corresponding to requests executed by a node amongst the sensor nodes	Section 3.6.1.3
Dynamic	Dynamic Random Allocation	DRA	Allocate sensor nodes at random upon arrival of a request.	Section 3.6.2.1
	Dynamic CPU Load Balanced Allocation	DCLBA	Allocate sensor nodes dynamically and balance the energy consumption at the CPU component amongst the sensor nodes.	Section 3.6.2.2
	Dynamic Data Load Balanced Allocation	DDLBA	Allocate sensor nodes dynamically and balance the energy consumption at the radio component due to the transmission of response messages corresponding to requests executed by a node amongst the sensor nodes	Section 3.6.2.3
	Balanced Metric Allocation	BMA	Balance the total energy consumption amongst sensor nodes.	Section 3.6.2.4
	Maximum Energy First	MEF	Allocate sensor nodes with higher value of available energy first.	Section 3.6.2.5

While describing the objective of the algorithm, the term balance is used to mean "equalize" as the algorithms focus on equalizing the total energy consumption or specific components of energy consumption of the sensor nodes. The performance of the

allocation algorithms proposed in this research is measured in terms of the minimum energy at a sensor node at the end of the simulation period, network lifetime, and the rate of energy drain at a sensor node. These performance measures have been described in Section 3.7 of Chapter 3.

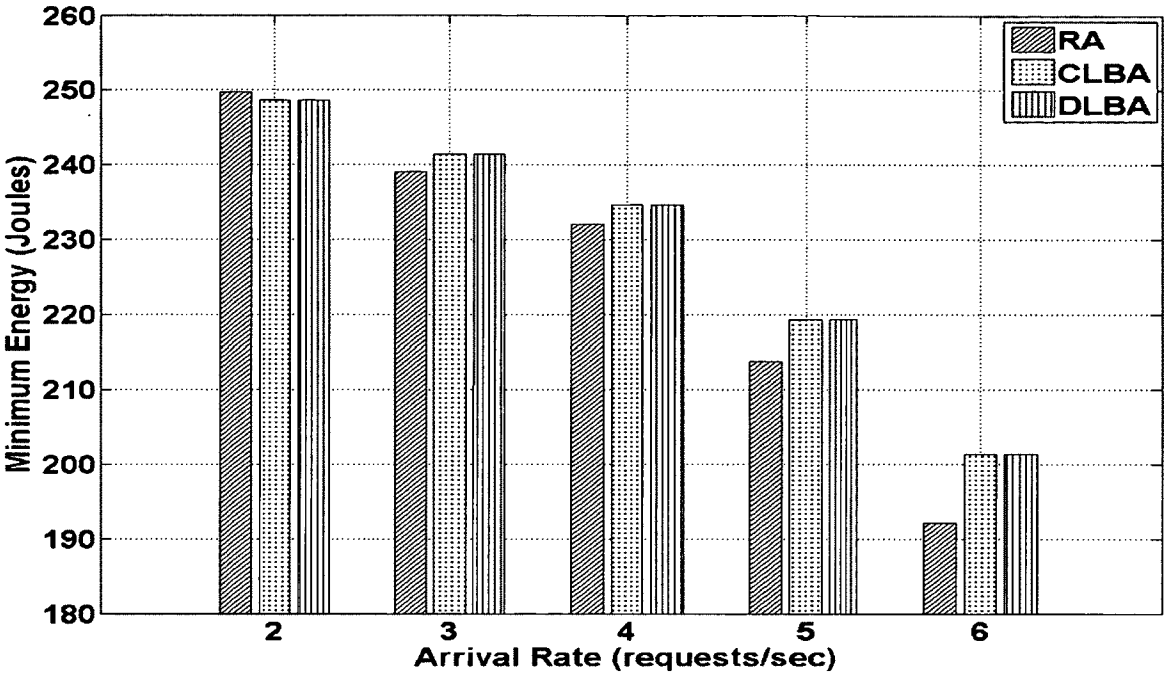
For the experiments with all the sensor nodes having equal amount of initial energy, network lifetime has been defined as the time when the energy of any sensor node falls to 500 Joules i.e. 50% of its initial energy value (the default value of initial energy of sensor nodes is 1000 Joules). For a set of experiments in which the sensor nodes have been assumed to have unequal amount of initial energy (initial energy distributed between 750 Joules and 1000 Joules), network lifetime has been defined as the time when the energy of any sensor node falls to 375 Joules i.e. 50% of minimum possible value of initial energy. The experiments were run long enough and repeated multiple times to obtain sufficiently small confidence intervals for the average values. For the experiments presented next, confidence intervals of ±2% (or less) for the performance metrics were obtained at a confidence level of 95% for all the algorithms except RA. For RA, confidence intervals of ±3% (or less) for the performance metrics were obtained at a confidence level of 95%. A part of the results presented in this chapter are included in [109].

5.1 Effect of Arrival Rate on the Performance of Algorithms

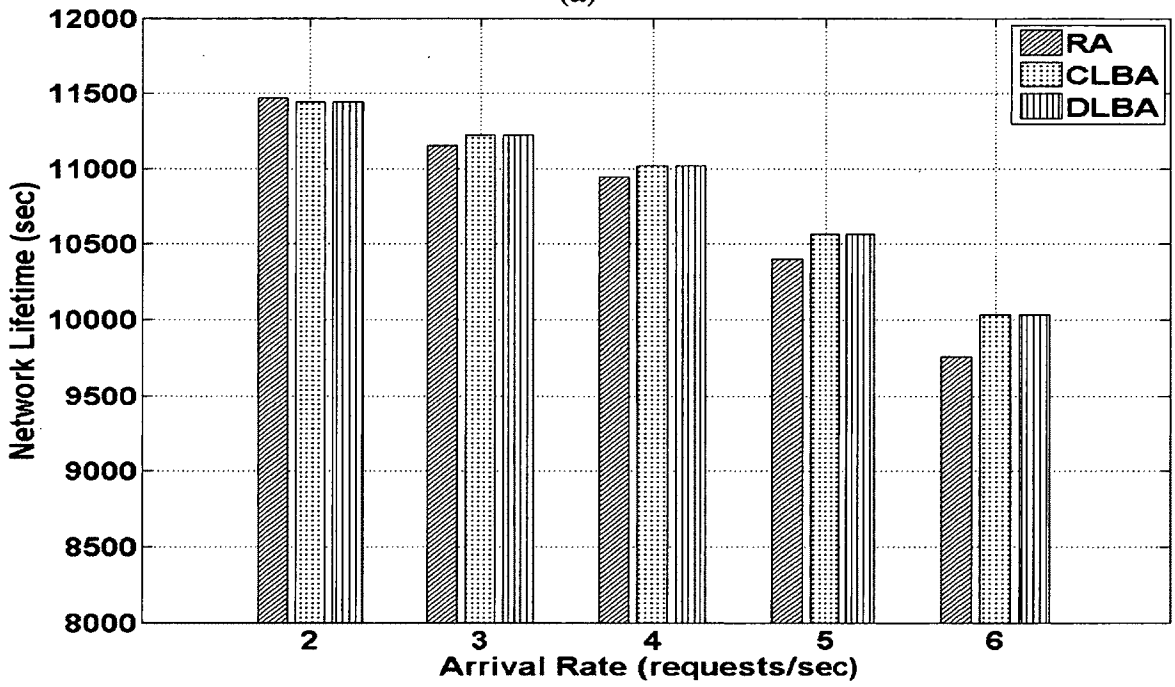
With this experiment, various system and workload parameters are held at their default values (given in Table 5-1) and only the arrival rate is varied.

5.1.1 Effect of Arrival Rate on the Performance of Static Allocation Algorithms

The performance of the static algorithms is shown in Figure 5-1. For all the algorithms, the minimum energy at a sensor node at the end of the simulation period decreases with the increase in the arrival rate (see Figure 5-1 (a)). This may be attributed to the fact that the number of messages being transmitted per unit time in the network increases with the increase in the arrival rate. The performance of the CLBA algorithm and the DLBA algorithm is superior to the RA algorithm. The minimum energy observed at a sensor node in case of the RA algorithm is lower as compared to the minimum energy observed at a sensor node in case of the CLBA algorithm or the DLBA algorithm (see Figure 5-1(a)). The node that demonstrates the minimum energy is a sensor node at a distance of 1 hop from the cluster head that has maximum energy consumption due to relaying of request messages from the cluster head to the respective sensors at higher hop distances. In case of CLBA and DLBA algorithms, this node is not allocated as there are 25 sensors at each hop and each application requires only 5 sensors at each hop. With the RA algorithm, this node may be allocated and therefore at higher arrival rates, RA performs slightly inferior to DLBA and CLBA. Also, with the RA algorithm, a sensor node may be allocated to requests from both the applications which further increases the uneven allocation of sensor nodes as compared to the CLBA and DLBA algorithms. At a high arrival rate, the RA algorithm also demonstrates a smaller network lifetime and a higher value of rate of energy drain (see Figure 5-1(b) and Figure 5-2 respectively).



(a)



(b)

Figure 5-1: Effect of Arrival Rate on the Performance of Static Allocation Algorithms

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

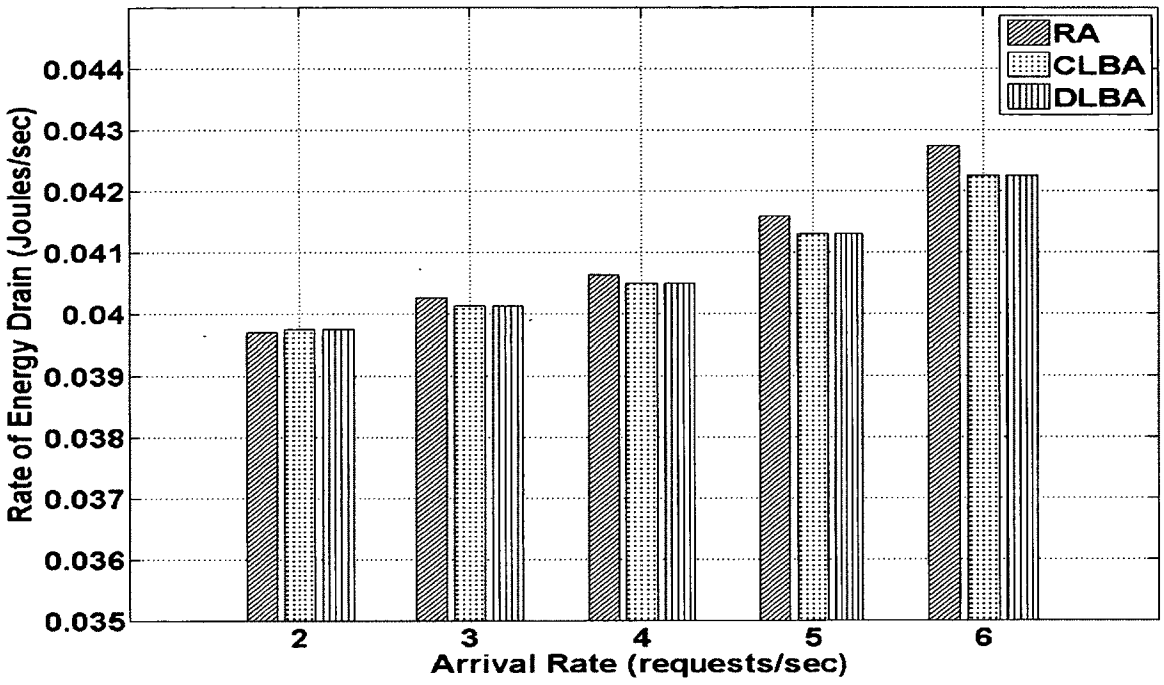


Figure 5-2: Effect of Arrival Rate on the Rate of Energy Drain of Static Allocation Algorithms

5.1.2 Effect of Arrival Rate on the Performance of Dynamic Allocation Algorithms

The effect of arrival rate on the performance of the dynamic allocation algorithms is shown in Figure 5-3. As in the case of static algorithms, for all the dynamic algorithms, the minimum energy measured at the end of the simulation period decreases with the increase in the arrival rate as the number of messages being transmitted per unit time in the network increases with the increase in the arrival rate. The BMA algorithm and the MEF algorithm demonstrate the best performance. The value of minimum energy observed at a sensor node is highest with the BMA algorithm and the MEF algorithm (see Figure 5-3(a)). The BMA algorithm focuses on balancing the total energy consumption amongst sensor nodes, both due to the radio component and the CPU component. The MEF algorithm allocates sensor nodes based on their current energy level which is the available energy at a sensor node when the allocation is done. The DCLBA algorithm and

the DDLBA algorithm focus on balancing the energy consumption only due to the CPU component and due to the transmission of response messages corresponding to requests processed by the node respectively. These algorithms do not consider the total energy consumption at the sensor nodes before allocating the sensor nodes to requests for applications. It seems that by using only a specific energy component, they demonstrate an inferior performance as compared to the BMA algorithm and the MEF algorithm. The DRA algorithm selects the sensor nodes at random and may allocate sensor nodes with higher energy consumption to requests for applications, decreasing their energy further. The data collected from the simulation demonstrates that the BMA algorithm allocates sensor nodes with higher energy consumption at the radio component lesser number of times as compared to sensor nodes with lower energy consumption at the radio component. The allocation of a sensor node affects its energy consumption at the CPU component and the BMA algorithm focuses on balancing the sum of energy consumption due to the radio component and the CPU component. Simulation data for a sensor node with a low value of Tx_Time (id = 5) and a sensor node with a high value of Tx_Time (id = 25) at an arrival rate of 5 requests/second is presented in Table 5-4. Tx_Time denotes the time the sensor node spends in transmission of messages. A1 and A2 denote the number of times the given sensor node is allocated to requests for Application 1 and Application 2 respectively. T1, T2, T3, T4, T5, and T6 are intervals in time at which the simulation data for the sensor nodes is collected and T1>T2>T3>T4>T5>T6. It may be noted that for a given arrival rate, the time sensor node with id = 25 spends in transmit state is greater than the time sensor node with id = 5 spends in transmit state. Tx_Time determines the energy consumption at the radio component of the sensor node. As all

nodes have equal amount of initial energy, Tx_Time also determines the current energy level at a sensor node.

Table 5-4: Simulation Data at Arrival Rate = 5 requests/sec

Sensor Node / Algorithm	Id = 5				Id = 25			
	Time	**Tx_Time**	**A1**	**A2**	**Time**	**Tx_Time**	**A1**	**A2**
DRA	T1	17.68291	1569	1519	T1	456.7798	1464	1459
	T2	19.64284	3054	2960	T2	557.2043	3000	2926
	T3	20.43272	4645	4496	T3	557.9926	4500	4446
	T4	21.22456	6038	6026	T4	558.7734	6038	6026
	T5	22.00072	7521	7523	T5	559.5559	7521	7523
	T6	22.78943	9051	9018	T6	560.3375	9051	9018
DCLBA	**Time**	**Tx_Time**	**A1**	**A2**	**Time**	**Tx_Time**	**A1**	**A2**
	T1	17.03150	1475	1548	T1	456.3546	1499	1523
	T2	18.86813	2978	3020	T2	553.3275	3001	2997
	T3	19.65801	4518	4528	T3	554.1158	4572	4474
	T4	20.44985	6034	6019	T4	554.8967	6094	5960
	T5	21.22601	7557	7508	T5	555.6791	7586	7478
	T6	22.01472	9101	9030	T6	556.4607	9122	9008
BMA	**Time**	**Tx_Time**	**A1**	**A2**	**Time**	**Tx_Time**	**A1**	**A2**
	T1	20.07947	1230	1268	T1	439.60	0	0
	T2	21.74925	7683	7518	T2	510.4596	0	0
	T3	22.53913	15378	15063	T3	511.2479	0	0
	T4	23.33097	22822	22657	T4	512.0288	0	0
	T5	24.10713	30335	30199	T5	512.8112	0	0
	T6	24.89584	38052	37810	T6	513.5928	0	0
MEF	**Time**	**Tx_Time**	**A1**	**A2**	**Time**	**Tx_Time**	**A1**	**A2**
	T1	19.66215	1197	1253	T1	441.0619	0	0
	T2	21.58901	7779	7580	T2	508.3934	0	0
	T3	22.37889	15474	15125	T3	509.1817	0	0
	T4	23.17073	22918	22719	T4	509.9626	0	0
	T5	23.94689	30431	30261	T5	510.7450	0	0
	T6	24.73560	38148	37872	T6	511.5266	0	0

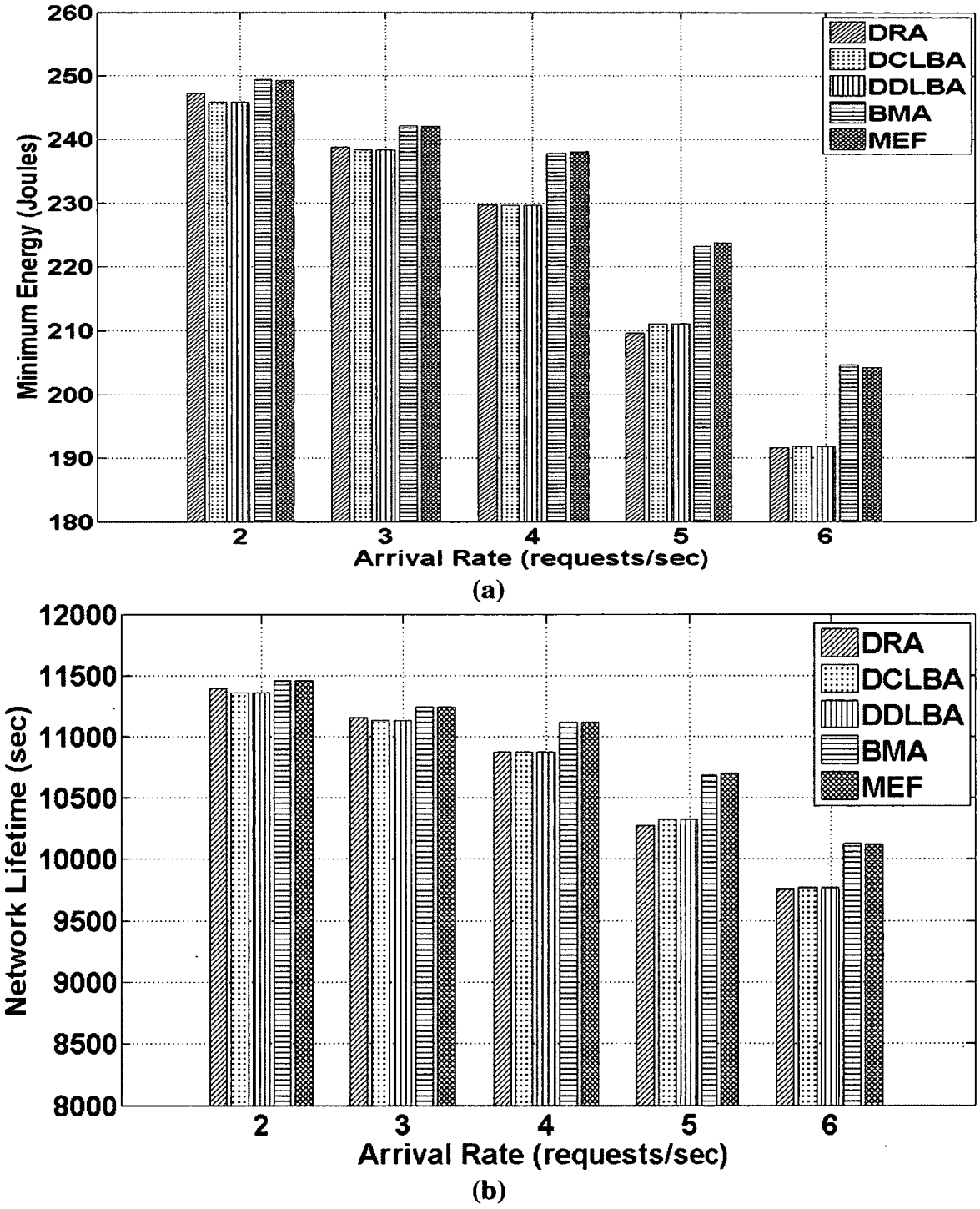


Figure 5-3: Effect of Arrival Rate on the Performance of Dynamic Allocation Algorithms

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

It can be seen from Table 5-4 that with the BMA algorithm and the MEF algorithm, the sensor node with higher value of Tx_Time is not allocated to requests for Application 1 and Application 2 (see columns for A1 and A2 for the BMA and MEF algorithms in Table 5-4). The DCLBA algorithm, which aims to balance the energy consumption only at the CPU component, allocates requests for Application 1 and Application 2 uniformly amongst the sensor nodes irrespective of the energy consumption at the radio component. Similar behaviour is observed with the DDLBA algorithm. With DRA, each node has equal probability of being allocated irrespective of its total energy consumption or current value of energy.

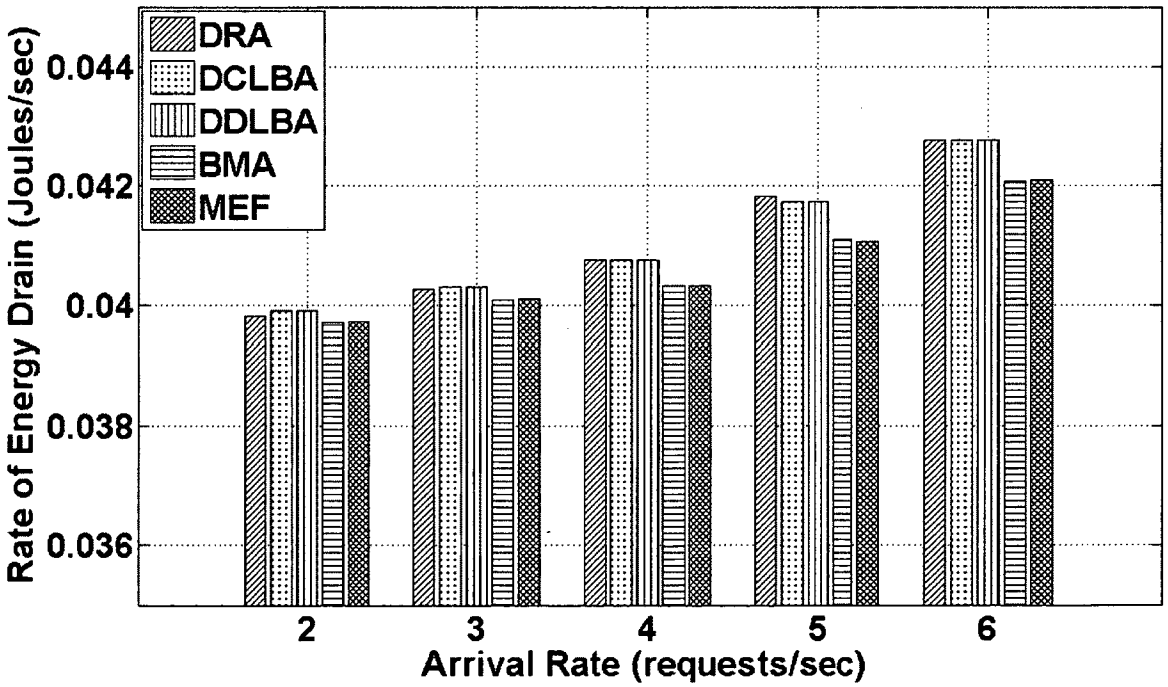


Figure 5-4: Effect of Arrival Rate on the Rate of Energy Drain of Dynamic Allocation Algorithms

The minimum energy observed at a sensor node with the BMA algorithm and the MEF algorithm is higher as compared to the minimum energy observed with DCLBA, DDLBA, and DRA (see Figure 5-3(a)). The BMA algorithm and the MEF algorithm also

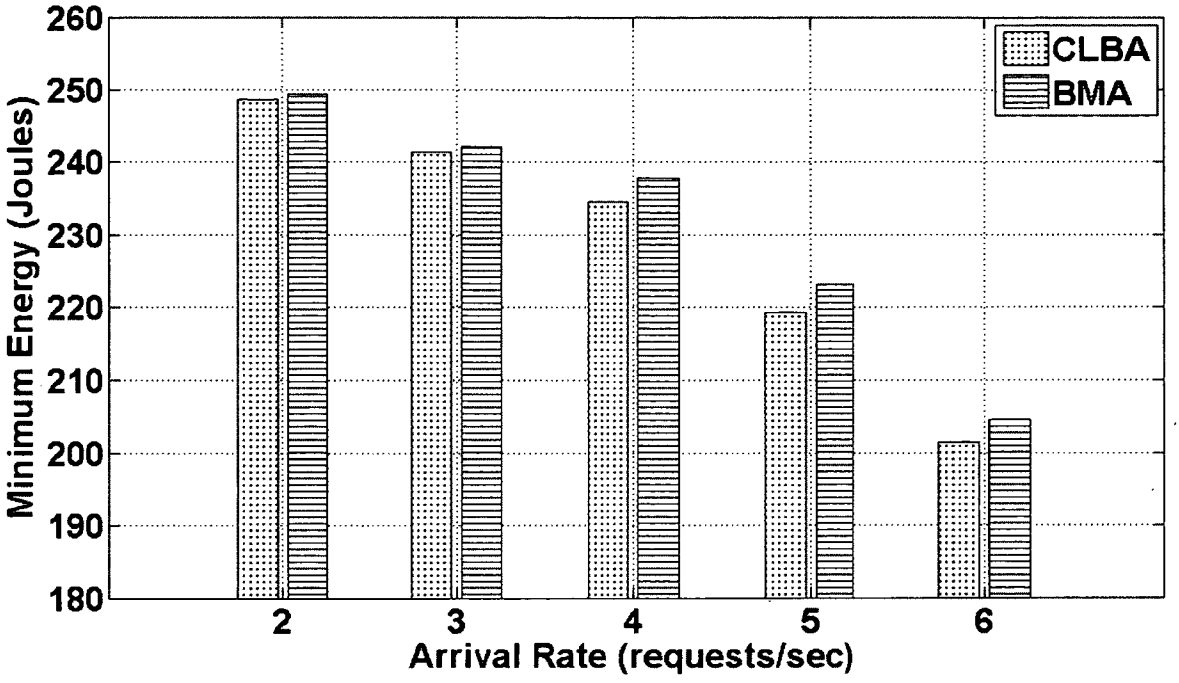
demonstrate higher values of network lifetime (see Figure 5-3(b)). As discussed in Section 3.7, network lifetime is the time when the energy of any node falls below a predefined threshold. The predefined threshold is 500 Joules for this experiment (50% of the initial value of energy of sensor nodes). As the BMA algorithm and the MEF algorithm focus on balancing the energy amongst the sensor nodes, they also demonstrate lower values of rate of energy drain for a sensor node (see Figure 5-4).

Note that the trend captured in the minimum energy graphs and the corresponding rates of energy drain graphs is similar to one another. Moreover, the rate of energy drain can be derived from the minimum energy at a sensor node. Thus, no further plots of rate of energy drain are provided.

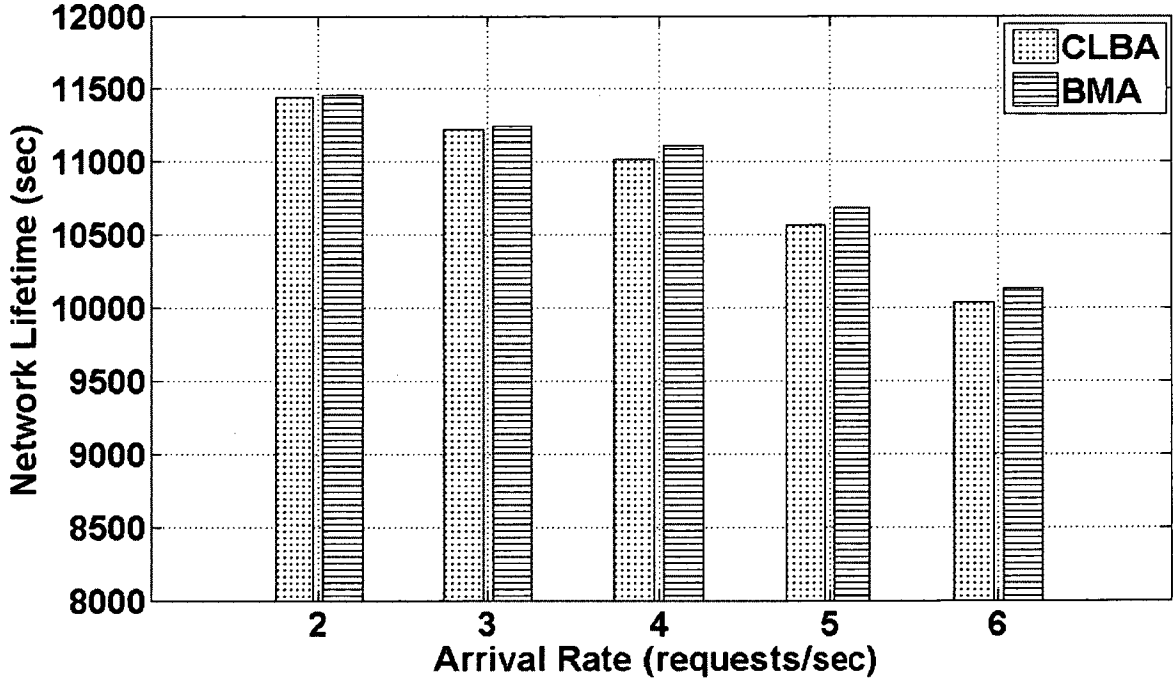
5.1.3 Comparison of Static and Dynamic Allocation

The dynamic allocation algorithms: the BMA algorithm and the MEF algorithm demonstrate the best performance among all static and dynamic algorithms. These algorithms allocate sensor nodes to job requests corresponding to an application request dynamically upon arrival of an application request at the proxy. BMA aims to balance the energy consumption both due to the CPU component and the radio component amongst all sensor nodes. MEF allocates sensor nodes to requests based on the current energy at the sensor nodes. DRA, DDLBA, and DCLBA demonstrate an inferior performance as they allocate sensor nodes to job requests without explicitly considering the energy level of the sensor nodes or their total energy consumption. This behaviour is demonstrated from the data collected corresponding to two sensor nodes presented in Table 5-4. From the table it can be seen that BMA and MEF do not allocate the sensor node with higher

value of Tx_Time (id = 25) whereas DRA and DCLBA allocate this node to requests for Application 1 and Application 2. RA, CLBA, and DLBA, which are the static counterparts of the DRA, DCLBA, and the DDLBA algorithms respectively, perform better than their dynamic counterparts. The static allocation algorithms allocate sensor nodes to applications and the allocation does not vary with each arriving application request. As each application requires only 5 sensor nodes out of the available 25 nodes at each hop distance, the sensor nodes with higher energy consumption are not allocated to requests for applications. For DRA, DCLBA, and DDLBA, the sensor nodes are allocated upon arrival of each application request at the proxy. DRA allocates the nodes at random with each node having an equal probability of being selected. Therefore, DRA also allocates the sensor nodes with high energy consumption at the radio component. DCLBA and DDLBA focus on balancing energy consumption only due to the CPU component or the radio component due to the transmission of response messages corresponding to requests processed by a node respectively (see Table 5-4). Therefore, they allocate all sensor nodes uniformly irrespective of their total energy consumption. The performance comparison of CLBA and BMA, which demonstrate the best static and dynamic performance respectively, is shown in Figure 5-5. DLBA provides a comparable performance to CLBA and MEF performs comparable to BMA. However, bars for only two algorithms are presented on the graph, one in the category of static algorithms and one in the category of dynamic algorithms. This is applicable for other figures in the chapter where best static performance is compared with the best dynamic performance. BMA performs better than CLBA by providing a better load distribution amongst the sensor nodes comprising the network.



(a)



(b)

Figure 5-5: Performance Comparison of Static and Dynamic Allocation for Varying Arrival Rate

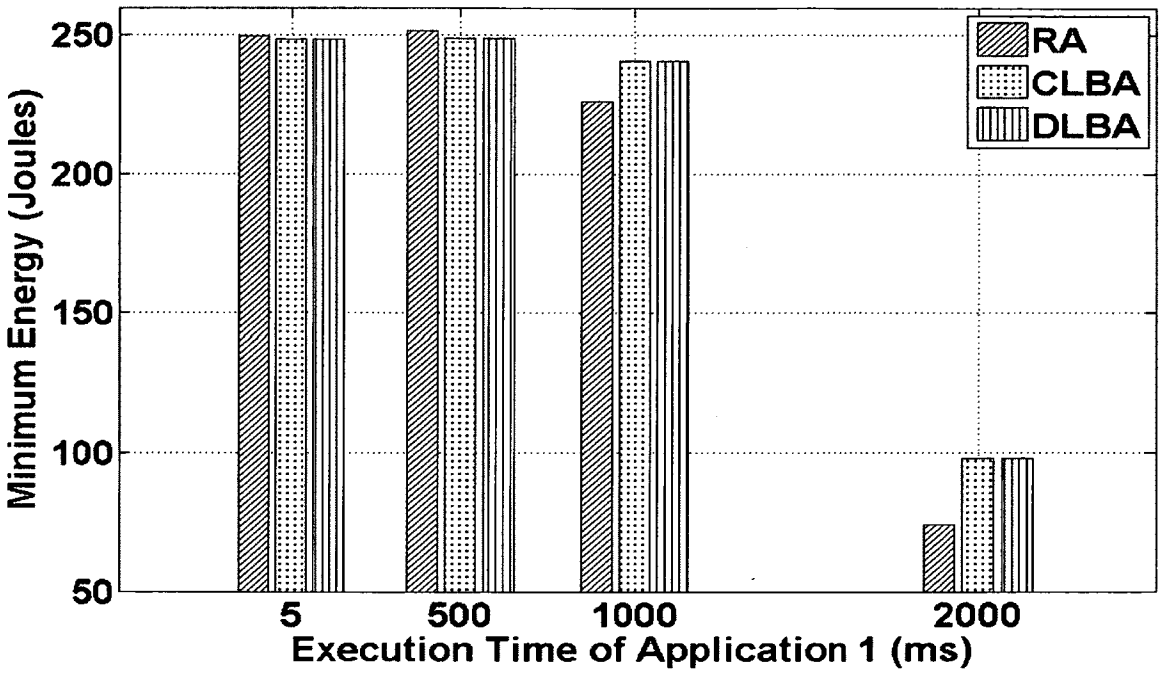
(a) Minimum Energy at a Sensor Node (b) Network Lifetime

5.2 Effect of Execution Time of Application on the Performance of Allocation Algorithms

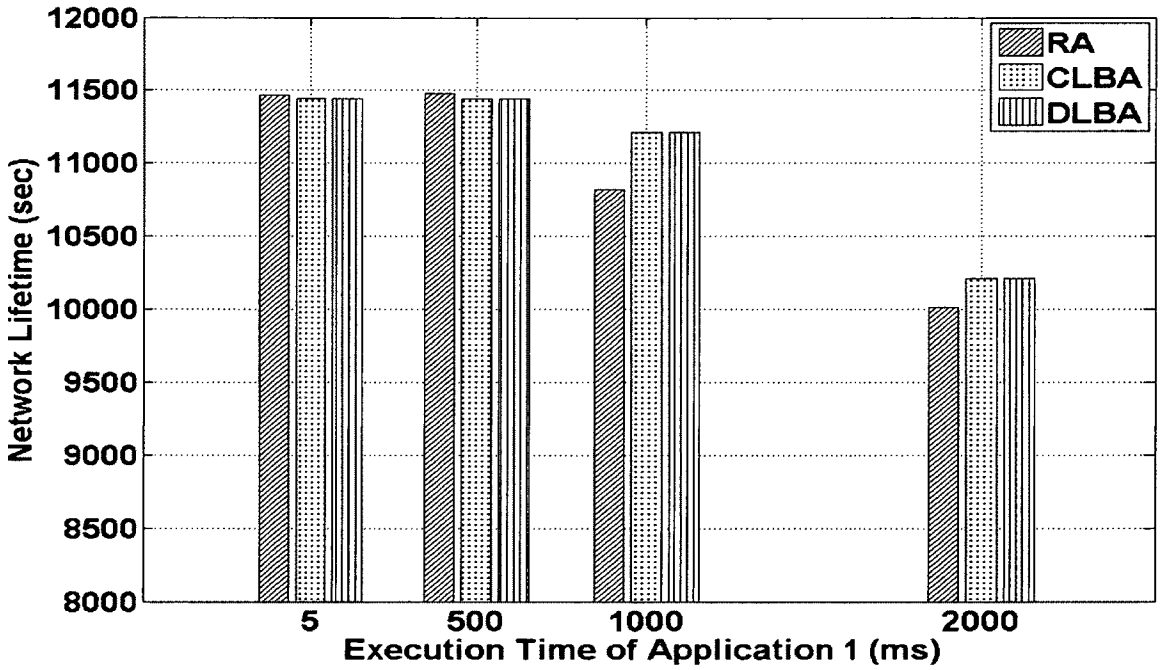
In this experiment, all system and workload parameters are held at the default values and only the execution time of Application 1 is varied. Execution time of only Application 1 is varied so that both the applications have different characteristics than one another. Another scenario where an experiment is performed with five applications with different characteristics is presented in Section 5.7.

5.2.1 Effect of Execution Time on the Performance of Static Allocation Algorithms

The effect of execution time of Application 1 on the performance of static allocation algorithms is shown in Figure 5-6. It can be observed from Figure 5-6 (a) that the minimum energy at a sensor node at the end of the simulation period decreases for higher values of execution time of Application 1. At higher values of execution time of Application 1, the energy consumption due to the CPU component is high for nodes that are allocated to Application 1. Hence, the rate of energy drain is high for such a node and it has low value of minimum energy at the end of the simulation period. The RA algorithm provides the worst performance because due to random allocation, a sensor node that is allocated to execute the requests for Application 1 may also be allocated to execute requests of Application 2. The CLBA and DLBA algorithms aim to balance the energy consumption at the CPU component and the radio component respectively. Therefore, CLBA and DLBA allocate different nodes to Application 1 and Application 2.



(a)



(b)

Figure 5-6: Effect of Execution Time on the Performance of Static Allocation Algorithms

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

For higher value of execution time of Application 1, the network lifetime decreases for all algorithms (see Figure 5-6 (b)). The simulation data for the sensor nodes that demonstrate minimum energy at the end of the simulation period for the CLBA algorithm and the RA algorithm is shown in Table 5-5 and Table 5-6 respectively. Tx_Time denotes the time the radio component of a sensor node spends in the transmit state and CPU_Time denotes the time the CPU component of a sensor node will spend in the active state for the allocated requests. A1 and A2 denote the number of times the given sensor node is allocated to requests for Application 1 and Application 2 respectively. It can be seen from Table 5-5 that the sensor node which is allocated to requests for Application 1 and which demonstrates minimum energy at the end of the simulation period is not allocated to requests for Application 2. In case of RA, the sensor node which demonstrates minimum energy at the end of the simulation period is allocated to both the requests for Application 1 and Application 2 (see Table 5-6). Such a node demonstrates a higher value of Tx_Time and CPU_Time as compared to the Tx_Time and the CPU_Time demonstrated with the CLBA algorithm.

Table 5-5: Simulation Data for Sensor Node with Minimum Energy for CLBA

Sensor Node / Time	Id = 5			
	Tx_Time	CPU_Time	A1	A2
T1	10.384475	6102	3051	0
T2	20.64216	12084	6042	0
T3	35.31565	18080	9040	0
T4	36.27967	24186	12093	0
T5	37.05583	30196	15098	0
T6	37.84453	36246	18123	0

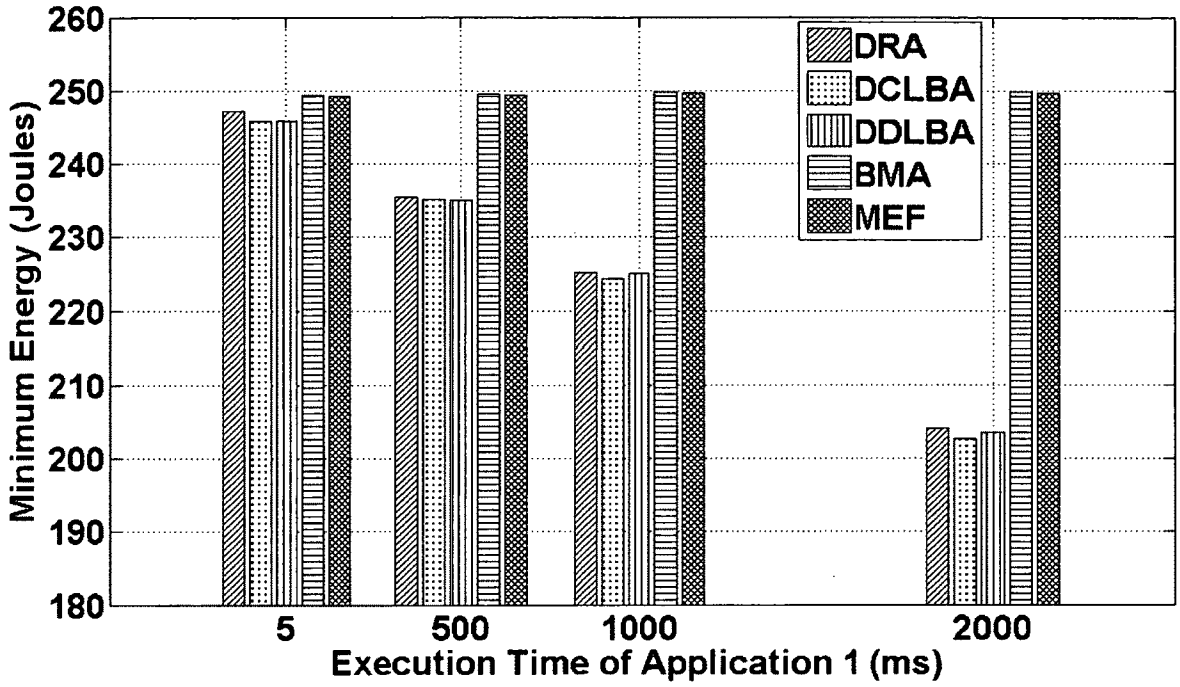
Table 5-6: Simulation Data for Sensor Node with Minimum Energy for the RA Algorithm

Sensor Node / Time	Id = 9			
	Tx_Time	CPU_Time	A1	A2
T1	19.89534	6109.445	3047	3089
T2	39.06081	12110.315	6040	6063
T3	64.08313	18101.19	9028	9038
T4	68.57196	24219.755	12080	11951
T5	69.35673	30284.555	15105	14911
T6	70.14623	36467.63	18189	17926

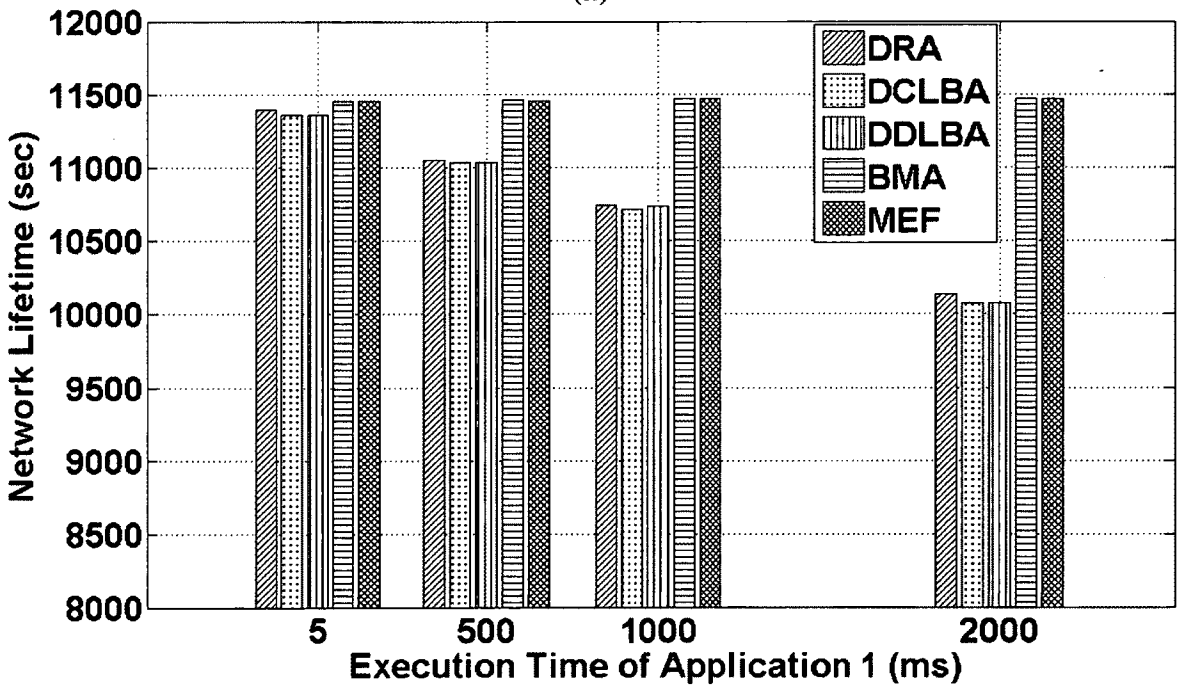
5.2.2 Effect of Execution Time on the Performance of Dynamic Allocation Algorithms

The effect of execution time of Application 1 on the performance of dynamic allocation algorithms is shown in Figure 5-7. Except for the BMA and the MEF algorithm, the minimum energy at a sensor node at the end of the simulation period decreases for all the other algorithms. The BMA algorithm takes allocation decisions based on the total energy consumption of the sensor nodes and the MEF algorithm takes allocation decisions based on the current energy level of the sensor nodes. Therefore, for these algorithms the minimum energy at a sensor node does not vary as the execution time of Application 1 is increased as the sensor node which has high energy consumption at the radio component due to the relaying of messages is not allocated to requests for Application 1. Hence, for such a node the energy consumption at the CPU component does not vary with the increase in the execution time for Application 1. For the DCLBA

and the DDLBA algorithm, the sensor nodes are allocated irrespective of their total energy consumption or their remnant energy level. The DCLBA and the DDLBA algorithms allocate sensor nodes to requests for applications on the basis of energy consumption due to the CPU component alone and the radio component alone respectively. The DRA algorithm allocates sensor nodes at random without considering the energy consumption or the current energy of the sensor nodes. Therefore DRA, DCLBA, and DDLBA allocate sensor nodes that have higher energy consumption or lower energy levels to requests and for these algorithms the minimum energy at a sensor node decreases with the increase in the execution time of Application 1 (see Figure 5-7 (a)). Also, the difference in the performance of the algorithms increases with the increase in the execution time of Application 1 as the energy consumption at the CPU component due to requests allocated for Application 1 increases with the increase in the execution time of Application 1. DCLBA, DDLBA, and DRA also demonstrate a lower network lifetime as compared to the BMA and the MEF algorithms (see Figure 5-7 (b)). The simulation data for the sensor that demonstrates minimum energy in case of the dynamic allocation algorithms for execution time of Application 1 = 2000 ms is given in Table 5-7. It can be seen from Table 5-7 that the sensor node which demonstrates a high value of Tx_Time is not allocated to requests for applications for BMA and is allocated only once for MEF. Tx_Time determines the energy consumption at the radio component of the sensor nodes which contributes towards the total energy consumption of the sensor nodes.



(a)



(b)

Figure 5-7: Effect of Execution Time on the Performance of Dynamic Allocation Algorithms

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

Table 5-7: Simulation Data for Sensor Node with Minimum Energy for Execution Time of Application 1 = 2000 ms

Sensor Node / Algorithm	Id = 25				
DCLBA	**Time**	**Tx_Time**	**CPU_Time**	**A1**	**A2**
	T1	171.01704	1221.265	609	653
	T2	337.40984	2422.25	1208	1250
	T3	426.909752	3621.07	1806	1814
	T4	427.690616	4843.985	2416	2397
	T5	428.473048	6057.285	3021	3057
	T6	429.254696	7292.295	3637	3659
BMA	**Time**	**Tx_Time**	**CPU_Time**	**A1**	**A2**
	T1	165.6986	0	0	0
	T2	326.456488	0	0	0
	T3	414.820744	0	0	0
	T4	415.601608	0	0	0
	T5	416.38404	0	0	0
	T6	417.165688	0	0	0
MEF	**Time**	**Tx_Time**	**CPU_Time**	**A1**	**A2**
	T1	165.577352	2	1	0
	T2	326.750568	2	1	0
	T3	414.891072	2	1	0
	T4	415.671936	2	1	0
	T5	416.454368	2	1	0
	T6	417.236016	2	1	0

5.2.3 Comparison of Static and Dynamic Allocation

Amongst both the static and dynamic allocation algorithms, the BMA algorithm and the MEF algorithm demonstrate the best performance. The static allocation algorithms demonstrate a much inferior performance as compared to BMA and MEF

especially at higher values of execution time of Application 1. Amongst the static algorithms, RA demonstrates the worst performance as a sensor node allocated to Application 1 may also be allocated to Application 2. The CLBA and the DLBA algorithms do not allocate the same sensor node to both the applications and hence perform better than the RA algorithm (see Table 5-5 and Table 5-6). The simulation data for the sensor node that demonstrates minimum energy at the end of the simulation period for the CLBA algorithm is shown in Table 5-8.

Table 5-8: Simulation Data for Sensor Node with Minimum Energy for CLBA for Execution Time of Application 1 = 2000 ms

Sensor Node / Algorithm	Id = 5				
CLBA	**Time**	**Tx_Time**	**CPU_Time**	**A1**	**A2**
	T1	10.384475	6102	3051	0
	T2	20.64216	12084	6042	0
	T3	35.31565	18080	9040	0
	T4	36.27967	24186	12093	0
	T5	37.05583	30196	15098	0
	T6	37.84453	36246	18123	0
BMA	**Time**	**Tx_Time**	**CPU_Time**	**A1**	**A2**
	T1	8.358495	1277.2800	637	656
	T2	16.59210	2530.4100	1262	1282
	T3	21.73140	5936.8950	2961	2979
	T4	22.52324	12055.459	6013	5892
	T5	23.29940	18120.259	9038	8852
	T6	24.08811	24303.335	12122	11867

From the table it can be seen that for CLBA which performs static allocation, the sensor nodes that get allocated to Application 1 have very high energy consumption at the

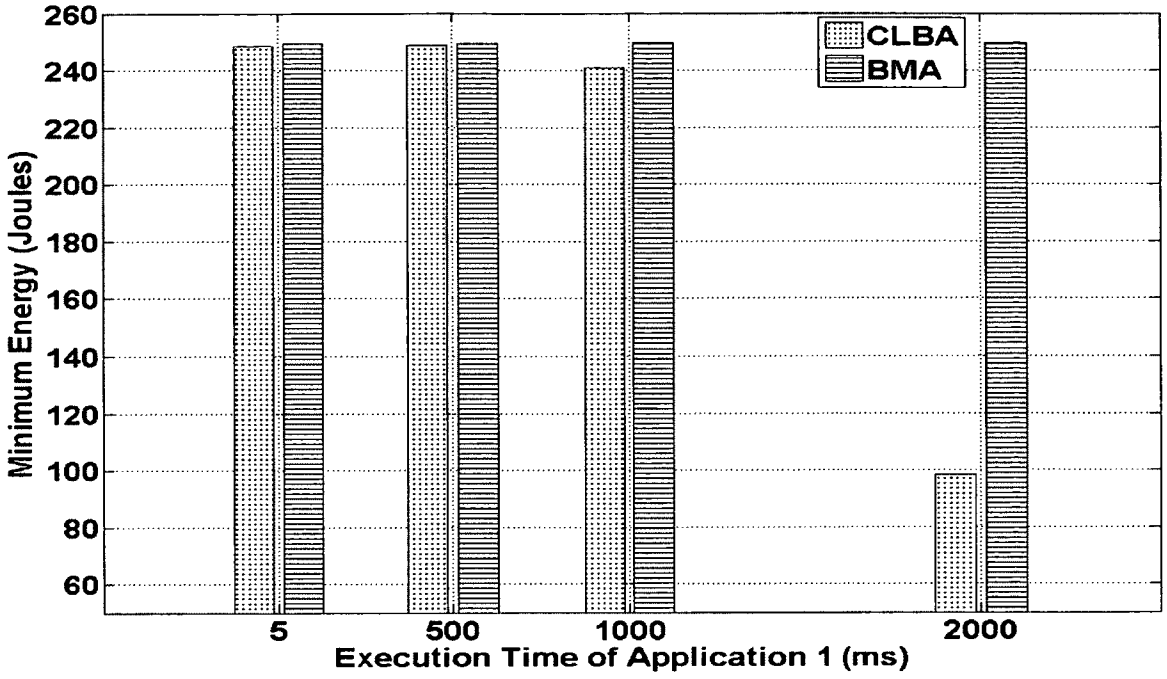
CPU component for higher values of execution time of Application 1 which leads to a higher rate of energy drain of the sensor nodes allocated to Application 1 and therefore a lower value of network lifetime. CLBA demonstrates the best performance amongst the static allocation algorithms and BMA performs the best among the dynamic allocation algorithms. The performance comparison of CLBA and BMA is presented in Figure 5-8. For execution time of Application 1 equal to 2000 ms, the difference in the minimum energy at a sensor node with CLBA and BMA algorithm is close to 60%. The superior performance of BMA over CLBA is clearly demonstrated. CLBA distributes load unevenly as compared to BMA.

5.3 Effect of Data Size on the Performance of Allocation Algorithms

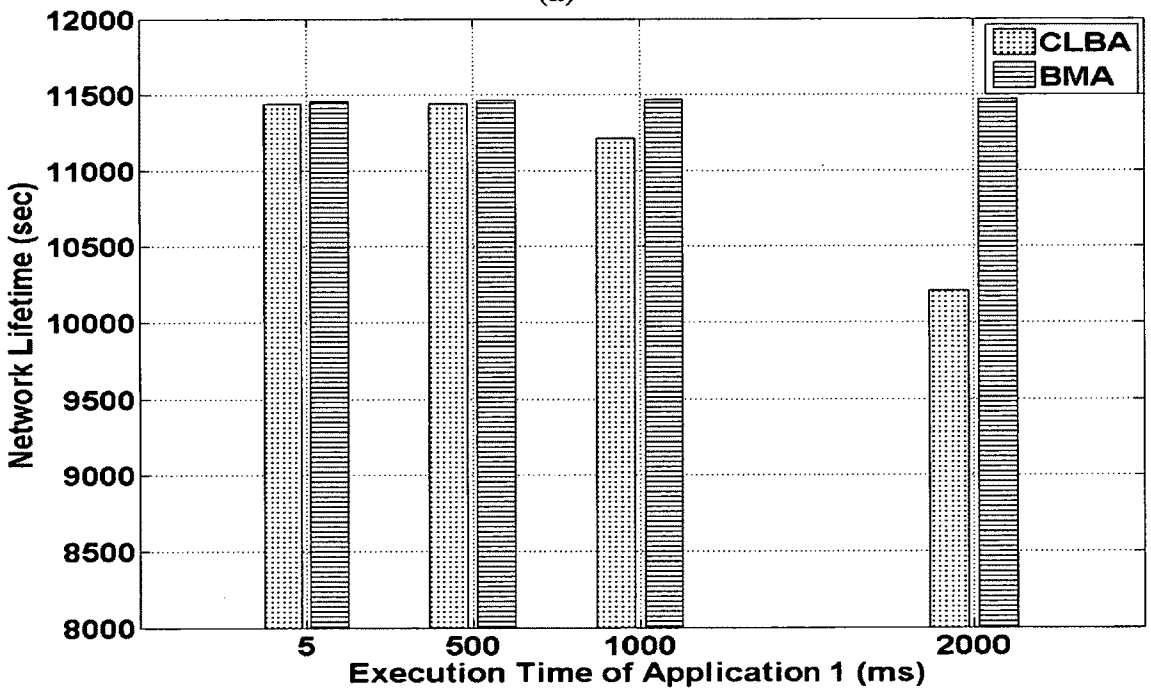
In this experiment, the data size of request and response messages for Application 1 is varied and all other system and workload parameters are held at their default values. Similar to the earlier experiment performed to study the effect of execution time of an application; data size of messages for Application 1 only is varied so that both Application 1 and Application 2 have different characteristics.

5.3.1 Effect of Data Size on the Performance of Static Allocation Algorithms

The performance of the static allocation algorithms as the data size is varied is shown in Figure 5-9. For all the algorithms, the minimum energy at a sensor node at the end of the simulation period decreases with the increase in the data size. This may be attributed to the fact that energy expended while transmitting a message is proportional to the size of the message. At higher values of data size for Application 1, the RA algorithm demonstrates a slightly better performance than the CLBA and the DLBA algorithm.



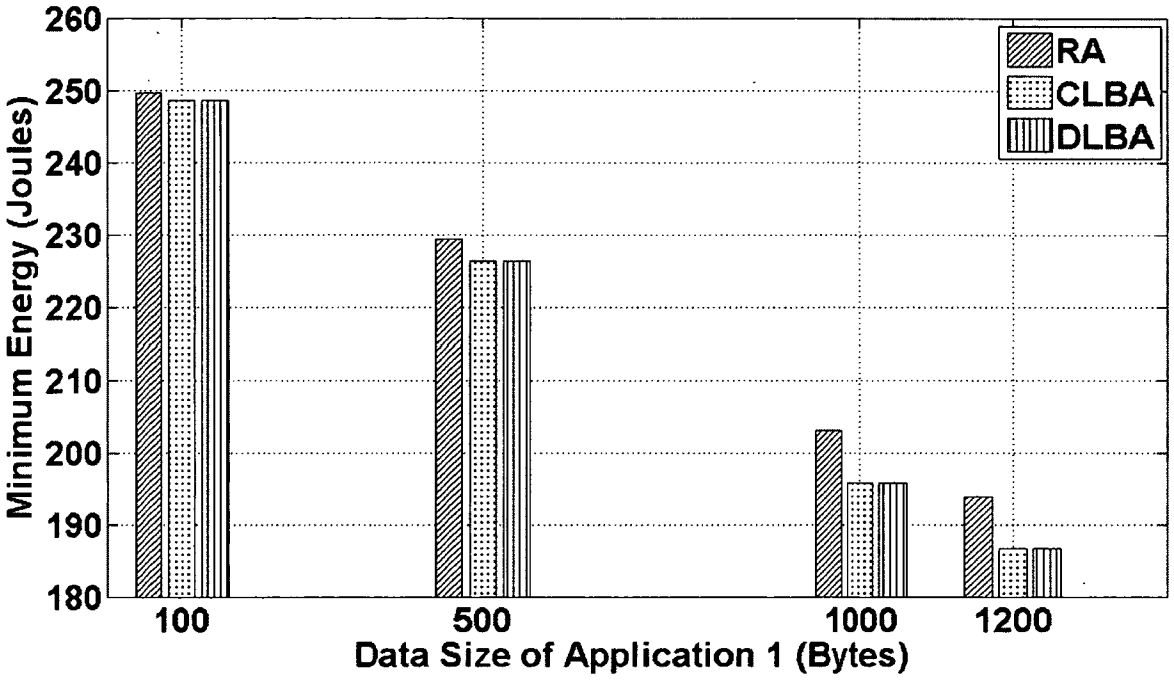
(a)



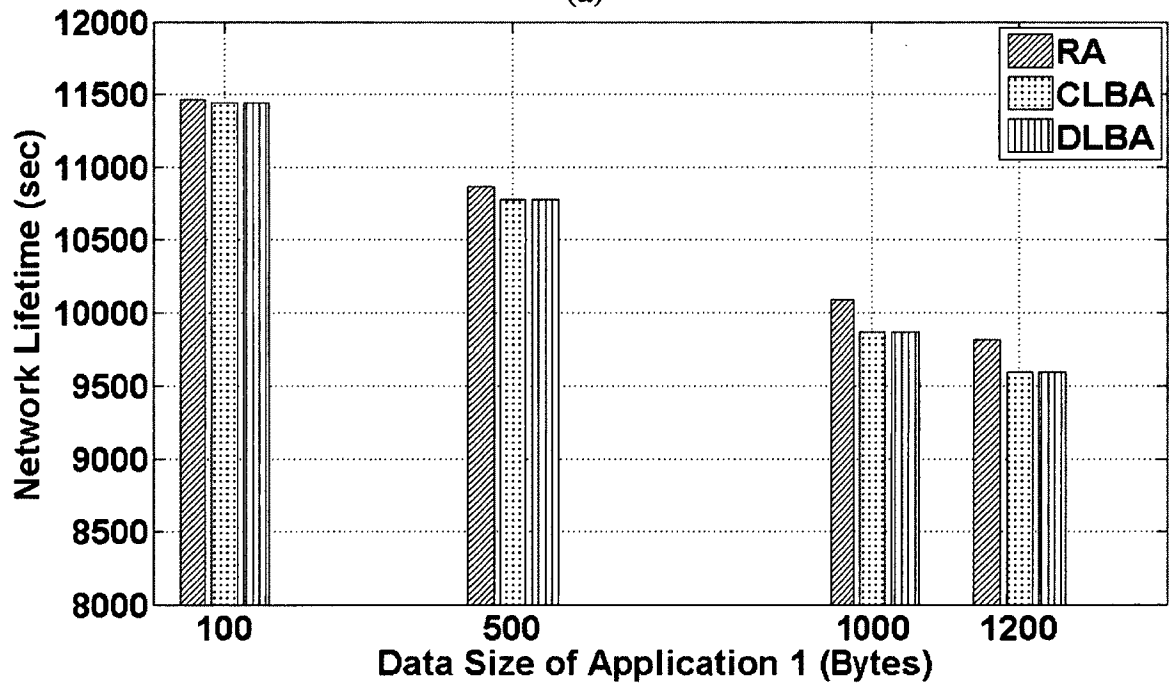
(b)

Figure 5-8: Performance Comparison of Static and Dynamic Allocation for Varying Execution Time of Application 1

(a) Minimum Energy at a Sensor Node (b) Network Lifetime



(a)



(b)

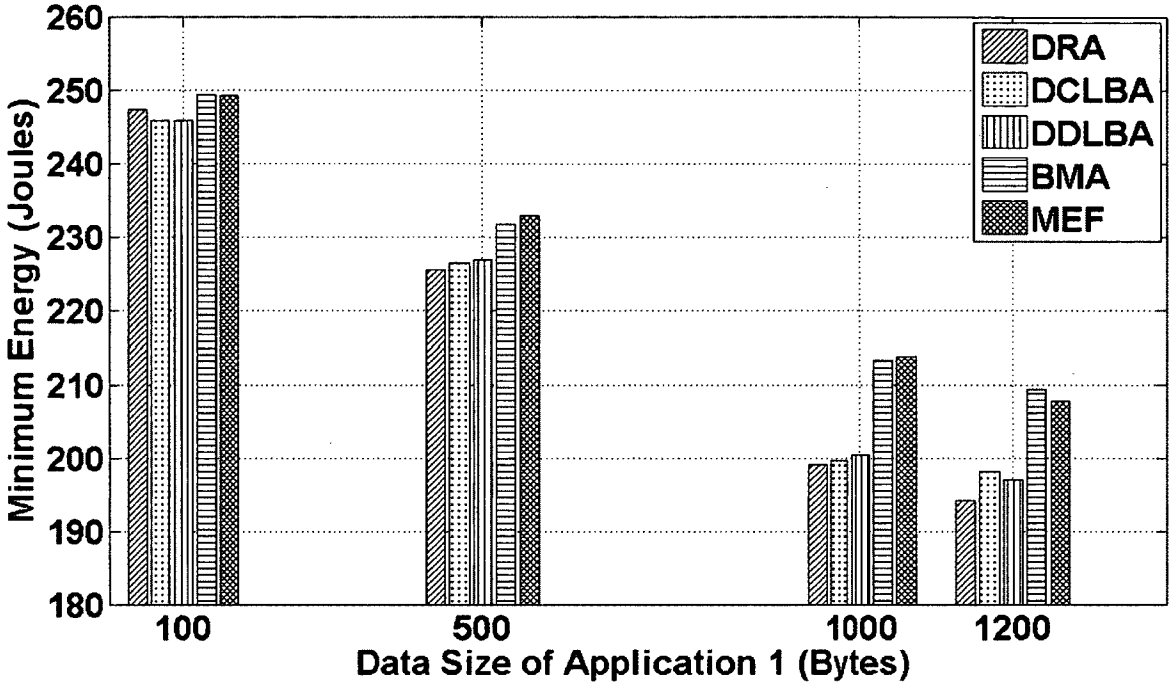
Figure 5-9: Effect of Data Size on the Performance of Static Allocation Algorithms (a) Minimum Energy at a Sensor Node (b) Network Lifetime

5.3.2 Effect of Data Size on the Performance of Dynamic Allocation Algorithms

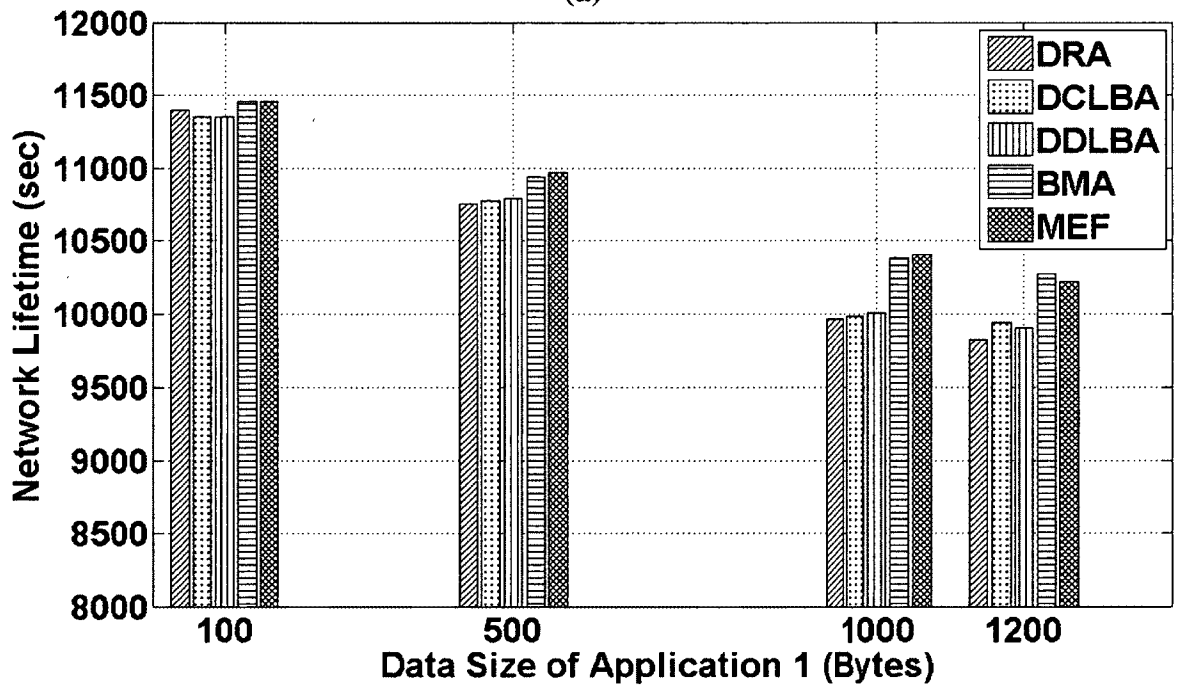
The performance of the dynamic allocation algorithms as the data size is varied is shown in Figure 5-10. The minimum energy at a sensor node at the end of the simulation period decreases for all the algorithms with an increase in the size of messages (see Figure 5-10 (a)). Also, the network lifetime decreases for all the algorithms with an increase in the size of messages (see Figure 5-10 (b)). Once again, the BMA algorithm and the MEF algorithm demonstrate the best performance as they allocate requests to sensor nodes based on their total energy consumption or value of current energy respectively. The DRA algorithm allocates requests at random. Also, the DCLBA and the DDLBA algorithms aim to balance only some of the components of energy consumption amongst the sensor nodes and hence allocate sensor nodes irrespective of their total energy consumption or the current level of energy at the sensor nodes. This phenomenon was earlier explained in detail in Section 5.1.2. As a result, the DCLBA algorithm, the DDLBA algorithm, and the DRA algorithm demonstrate an inferior performance as compared to the MEF and the BMA algorithm.

5.3.3 Comparison of Static and Dynamic Allocation

The BMA and the MEF algorithm demonstrate the best performance. As described in Section 5.1.2, the BMA algorithm balances the total energy consumption amongst all the sensor nodes. The MEF algorithm allocates sensor nodes based on their available energy. The performance comparison of RA and BMA, which provide the best performance amongst the static allocation algorithms and the dynamic allocation algorithms respectively, is shown in Figure 5-11.



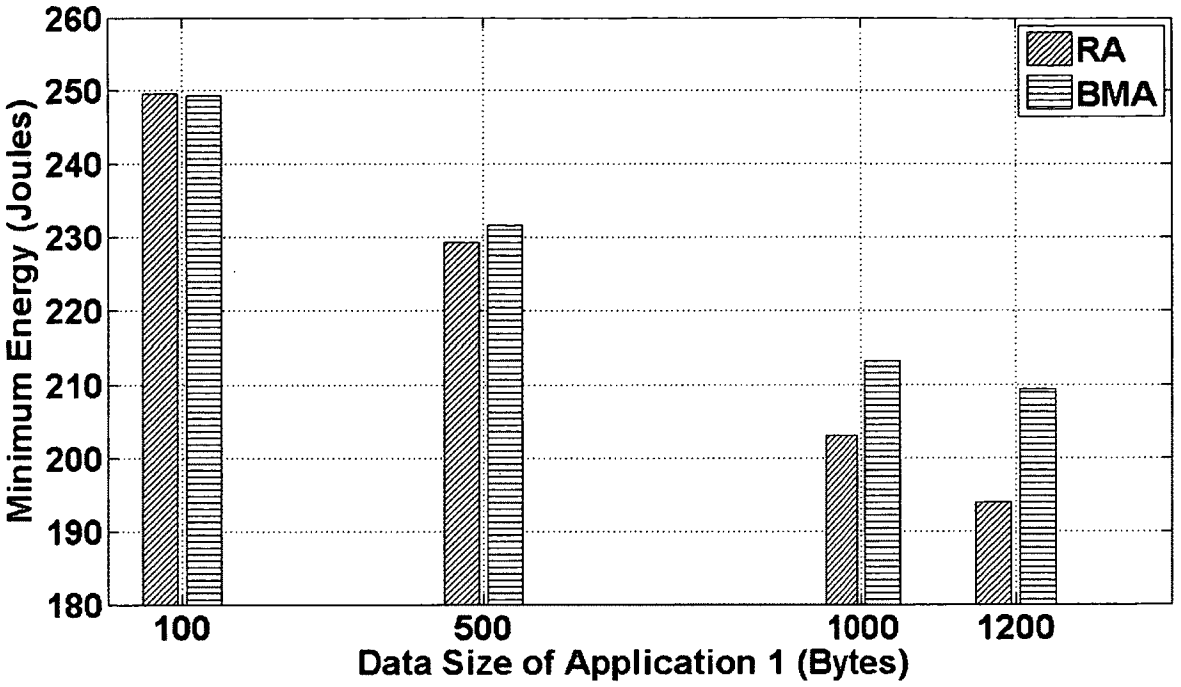
(a)



(b)

Figure 5-10: Effect of Data Size on the Performance of Dynamic Allocation Algorithms

(a) Minimum Energy at a Sensor Node (b) Network Lifetime



(a)

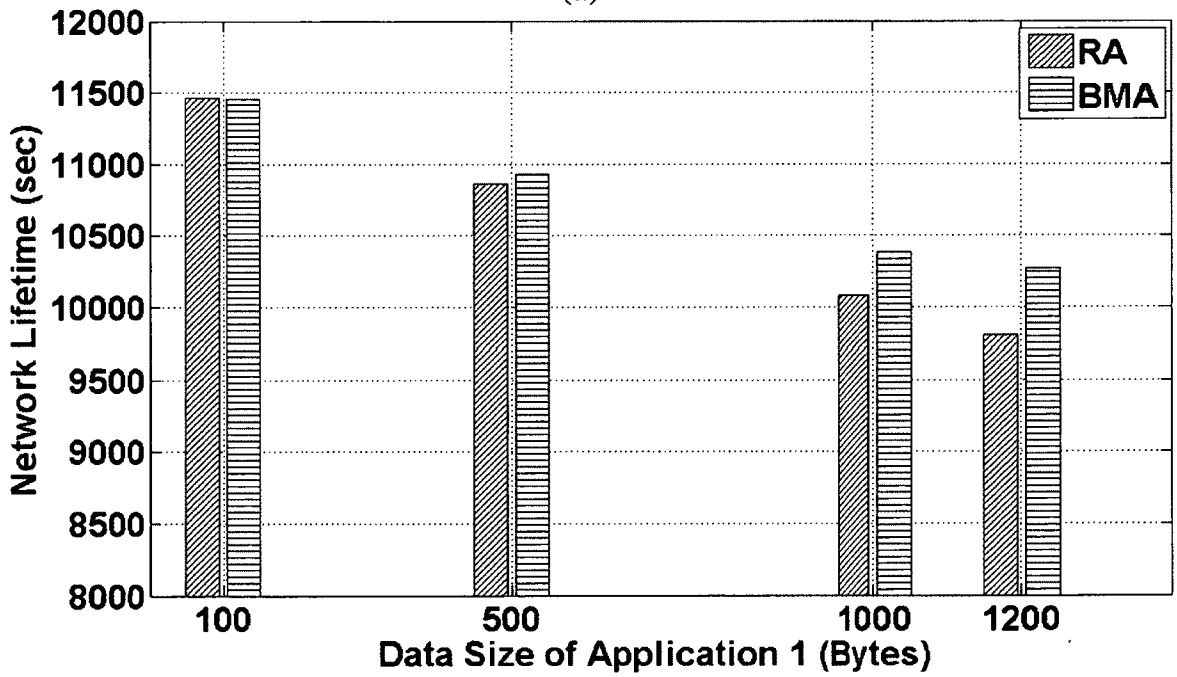


Figure 5-11: Performance Comparison of Static and Dynamic Allocation for Varying Data Size of Application 1

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

The static allocation algorithms cause an uneven distribution of requests amongst the sensor nodes comprising the WSN and hence demonstrate an inferior performance as

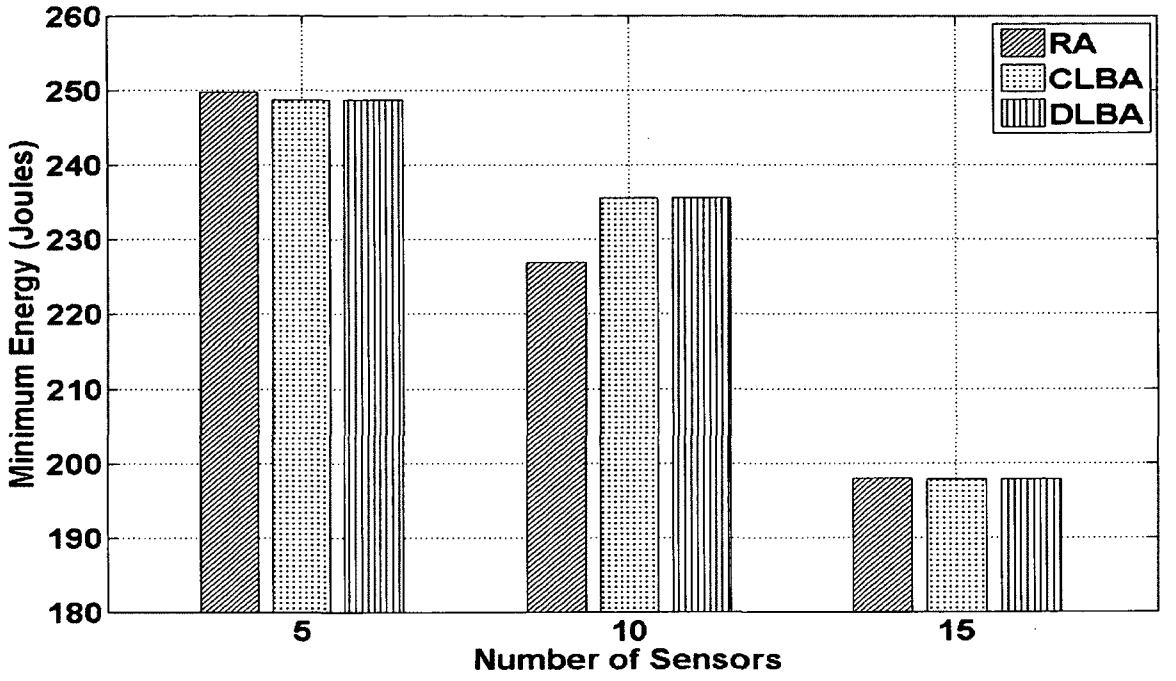
compared to BMA and MEF. With the dynamic allocation algorithms, requests get distributed to a larger number of sensors as compared to the static algorithms; therefore they provide a better load distribution. The difference in the performance of the messages increases with the increase in the size of the messages.

5.4 Effect of Number of Sensors on the Performance of Allocation Algorithms

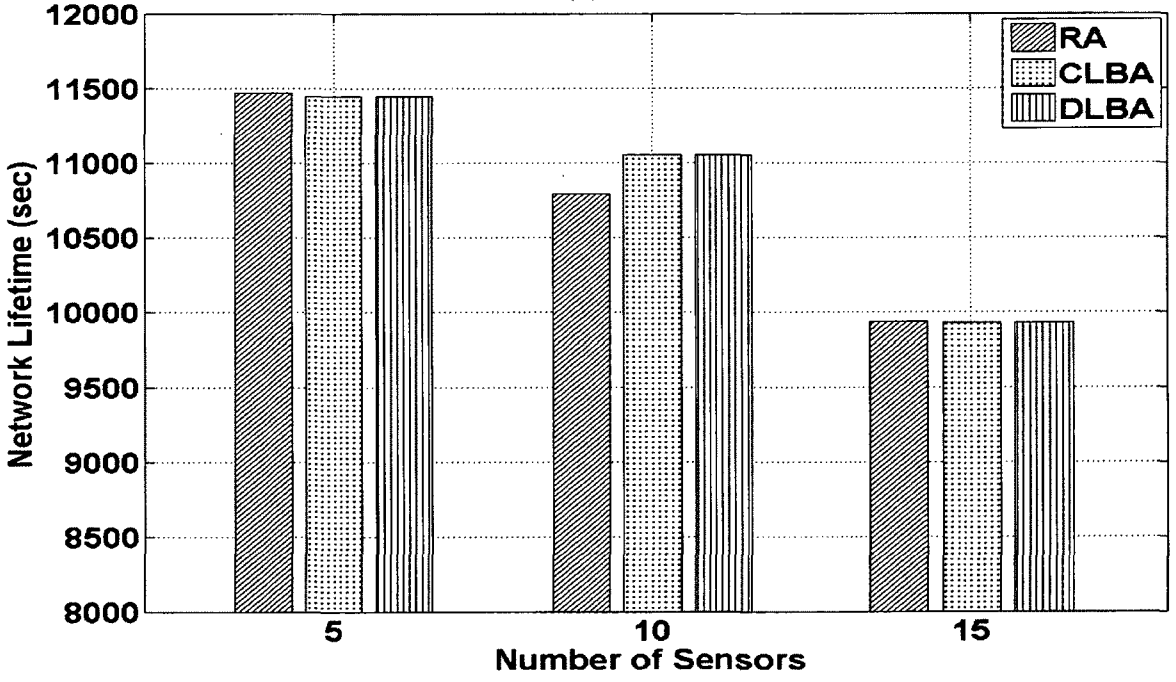
With this experiment, all the system and workload parameters are held at the default values and the number of sensors required by the applications at each hop distance is varied.

5.4.1 Effect of Number of Sensors on the Performance of Static Allocation Algorithms

For the default set of system parameters, both Application 1 and Application 2 require five sensors at each hop distance. With this experiment, the number of sensors required by Application 1 and Application 2 at each hop distance is increased from 5 to 10 and then from 10 to 15. The performance of the static allocation algorithms as the number of sensors required by the applications is increased is shown in Figure 5-12. For all the algorithms, as the number of sensors required by the applications is increased, minimum energy at a sensor node at the end of the simulation period decreases (see Figure 5-12(a)). With the increase in the number of sensors required by applications, the number of messages being transmitted in the WSN increases which causes a decrease in the energy of the sensors.



(a)



(b)

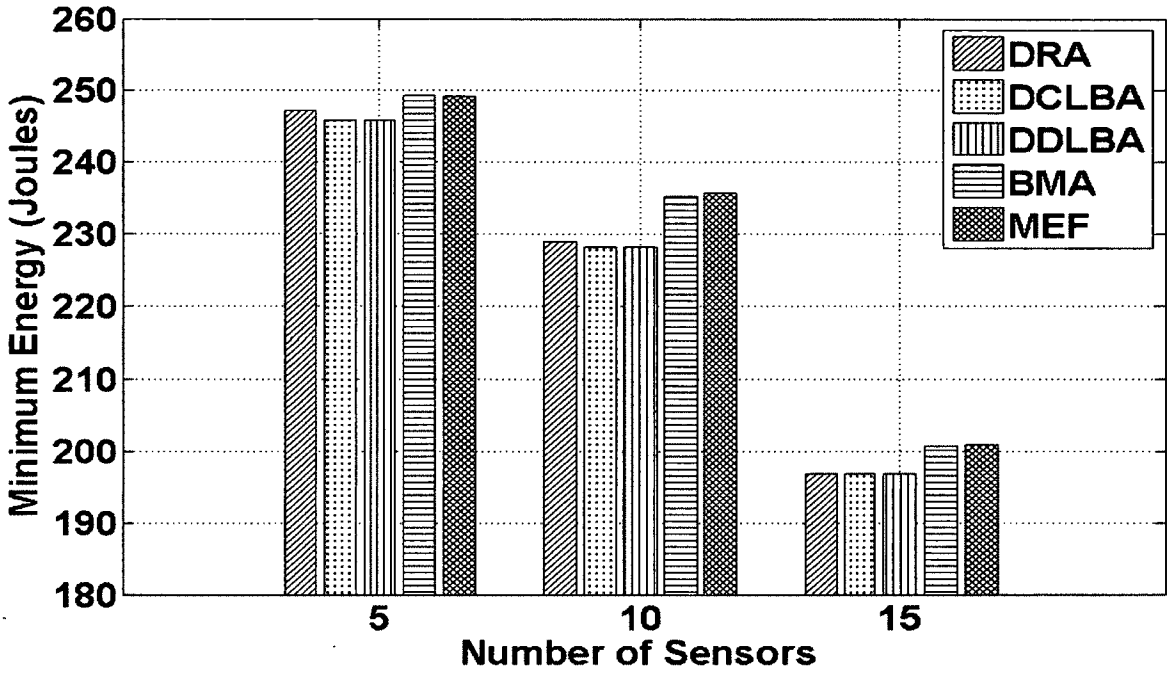
Figure 5-12: Effect of Number of Sensors on the Performance of Static Allocation Algorithms

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

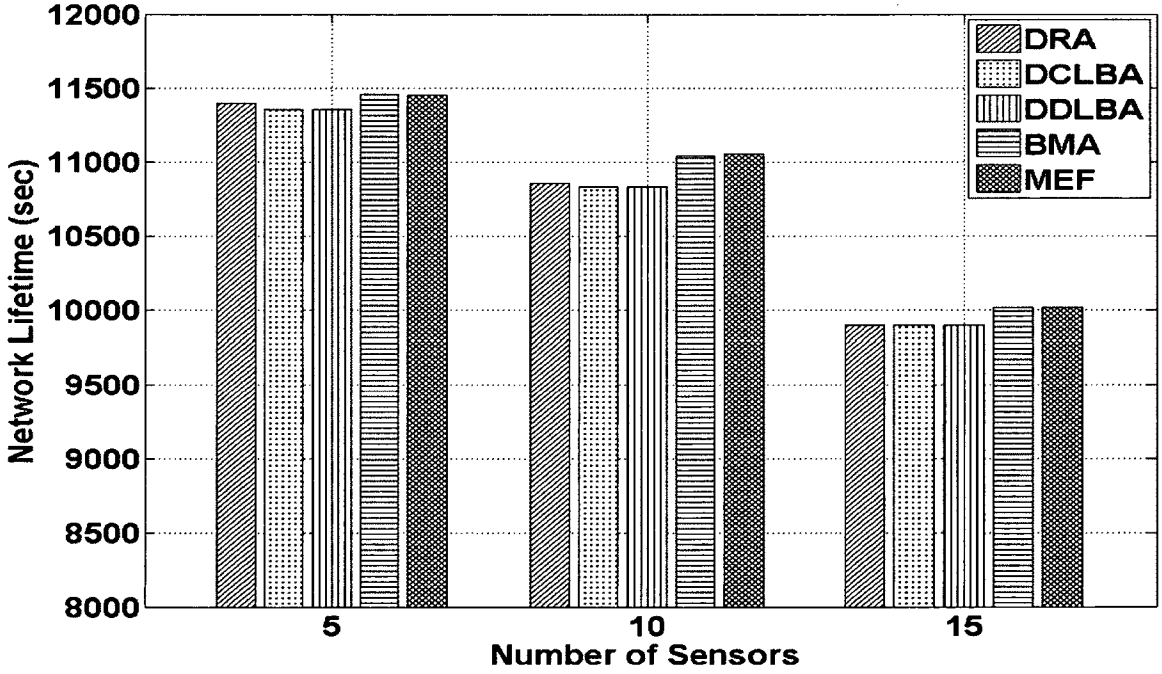
When the number of sensors required by each of the application is increased to 10 at each hop distance, CLBA and DLBA perform better than the Random Allocation algorithm as the node with maximum energy consumption is not allocated to any of the applications. A similar observation was discussed earlier in Section 5.1.1. For the RA algorithm, the probability of allocation of such a node increases as the number of sensors required by each of the application is increased from 5 to 10 and that accounts for the inferior performance of the RA algorithm. When the number of sensors required by the applications is increased to 15, the CLBA and the DLBA algorithms also allocate the sensor node with maximum energy consumption as each application requires 15 sensors at each hop distance and the number of available sensors at each hop is 25. Thus, RA, CLBA, and DLBA provide a comparable performance when the number of sensors required by the applications at each hop distance is increased to 15.

5.4.2 Effect of Number of Sensors on the Performance of Dynamic Allocation Algorithms

The performance of the dynamic allocation algorithms as the number of sensors required by the applications is increased is shown in Figure 5-13. As the number of sensors required by the applications increases, the number of messages being transmitted in the network increases. Once again, for all the algorithms, the minimum energy at the sensor node decreases with the increase in the number of sensors required by the applications. The BMA algorithm and the MEF algorithm perform better than the DRA algorithm, the DCLBA algorithm, and the DDLBA algorithm as they allocate sensor nodes based on their total energy consumption or current energy level.



(a)



(b)

Figure 5-13: Effect of Number of Sensors on the Performance of Dynamic Allocation Algorithms

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

5.4.3 Comparison of Static and Dynamic Allocation Algorithms

Amongst the dynamic allocation algorithms, the BMA algorithm and the MEF algorithm demonstrate the best performance. The CLBA and the DLBA algorithms provide the best performance amongst static allocation algorithms. The performance comparison of CLBA and BMA is shown in Figure 5-14. The CLBA algorithm provides a performance comparable to the BMA algorithm when the numbers of sensors required by the applications at each hop distance are 5 or 10. This is because the sensor node that has maximum energy consumption is not allocated for these cases. When each of the application requires 15 sensors at each hop, the sensor node with the maximum energy consumption is allocated for CLBA and DLBA. Hence, their performance is slightly inferior to the BMA algorithm and the MEF algorithm. The simulation data for the sensor node that demonstrates minimum energy at the end of the simulation period for Ns = 5 and Ns = 15 is presented in Table 5-9. Ns denotes the number of sensors required by the applications at each hop distance. From Table 5-9, it can be seen that when the number of sensors required by each of the applications is 5, CLBA does not allocate the sensor node that demonstrates minimum energy at the end of the simulation period. When the number of sensors required by each of the applications is increased to 15, CLBA allocates this node whereas BMA allocates this node only once due to its high value of total energy consumption.

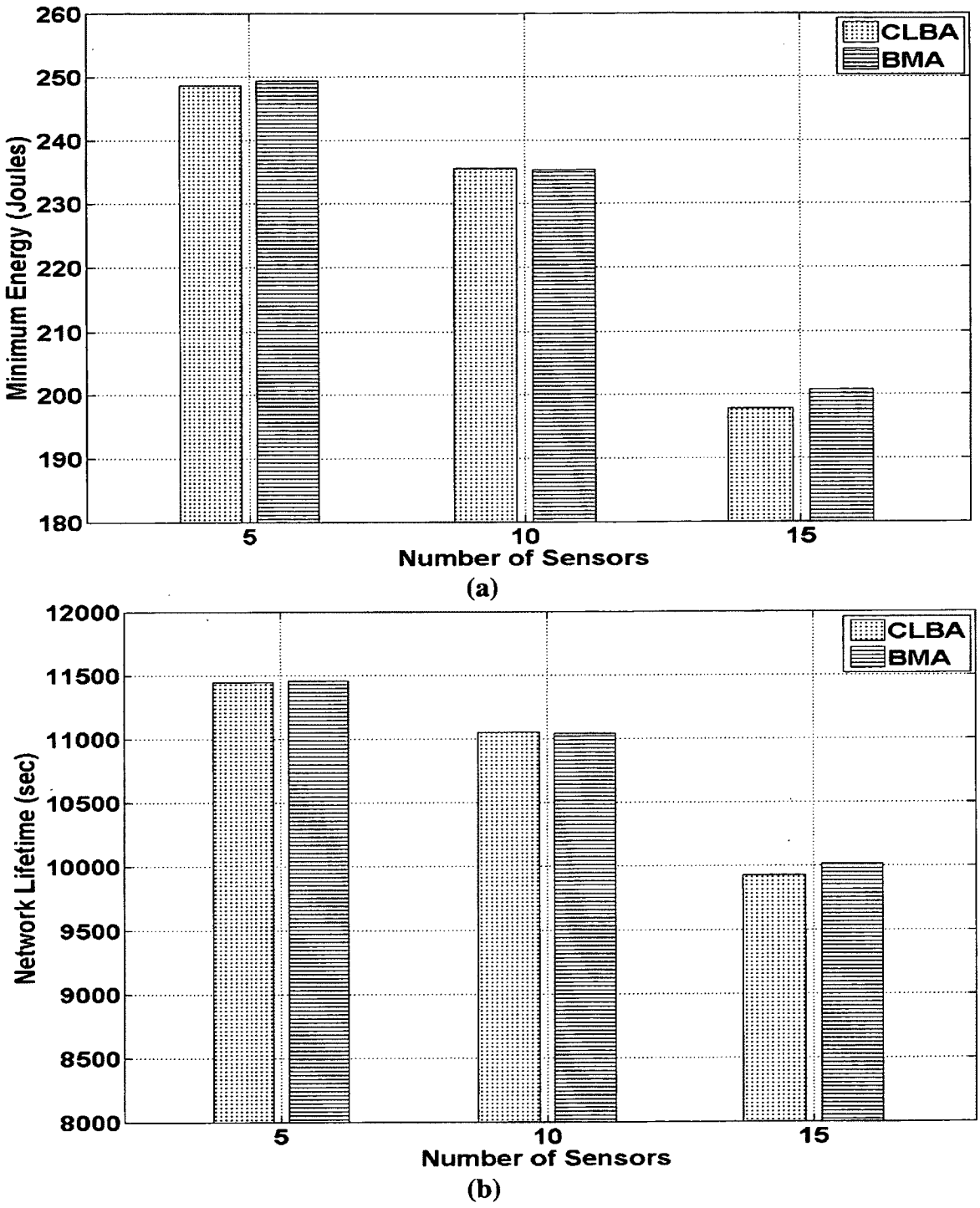


Figure 5-14: Performance Comparison of Static and Dynamic Allocation for Varying Number of Sensors Required by Applications

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

Table 5-9: Simulation Data for Sensor Node with Minimum Energy for Ns = 5 and Ns = 15

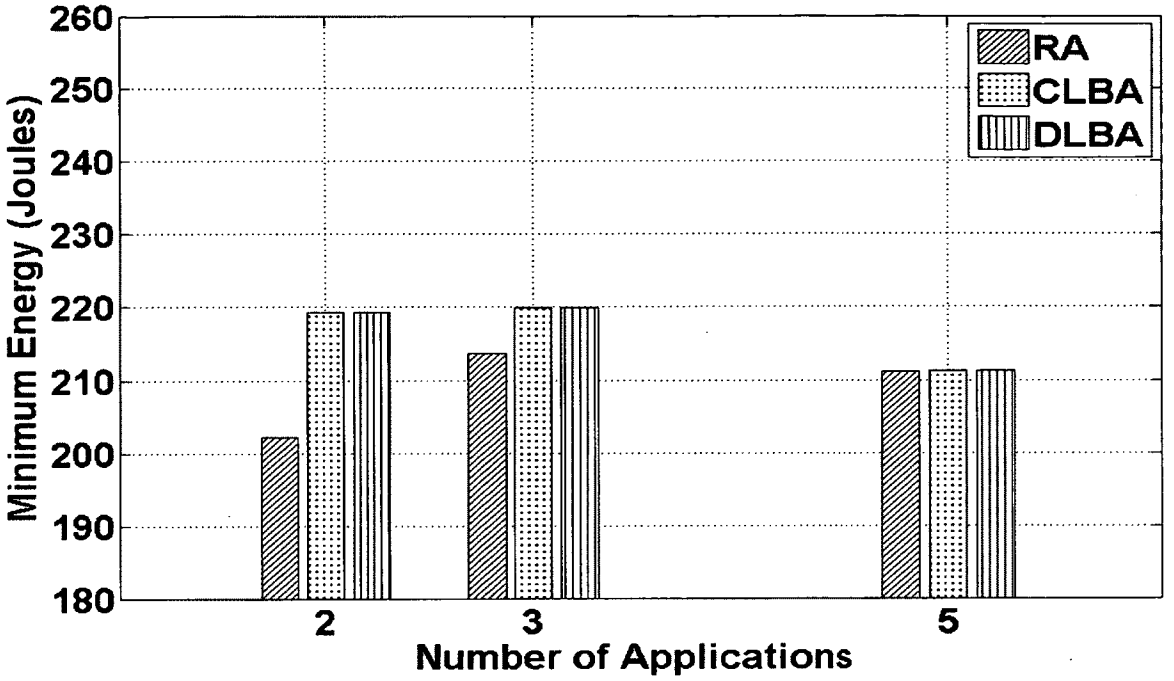
Ns / Algorithm	Ns = 5				Ns = 15			
CLBA	**Time**	**Tx_Time**	**A1**	**A2**	**Time**	**Tx_Time**	**A1**	**A2**
	T1	167.5245	0	0	T1	455.4982	0	3089
	T2	331.5282	0	0	T2	579.6138	0	6063
	T3	418.9406	0	0	T3	602.2372	0	9038
	T4	419.7215	0	0	T4	603.0181	0	11951
	T5	420.5039	0	0	T5	603.8005	0	14911
	T6	421.2856	0	0	T6	604.5821	0	17926
BMA	**Time**	**Tx_Time**	**A1**	**A2**	**Time**	**Tx_Time**	**A1**	**A2**
	T1	164.8575	0	0	T1	444.9317	1	0
	T2	324.9649	0	0	T2	568.8952	1	0
	T3	416.3800	0	0	T3	592.3578	1	0
	T4	417.1608	0	0	T4	593.1386	1	0
	T5	417.9433	0	0	T5	593.9211	1	0
	T6	418.7249	0	0	T6	594.7027	1	0
MEF	**Time**	**Tx_Time**	**A1**	**A2**	**Time**	**Tx_Time**	**A1**	**A2**
	T1	164.4678	0	0	T1	445.0649	1	0
	T2	325.0402	0	0	T2	569.1784	1	0
	T3	416.8125	0	0	T3	592.2121	1	0
	T4	417.5933	0	0	T4	592.9929	1	0
	T5	418.3758	0	0	T5	593.7754	1	0
	T6	419.1574	0	0	T6	594.5570	1	0

5.5 Effect of Number of Applications on the Performance of Allocation Algorithms

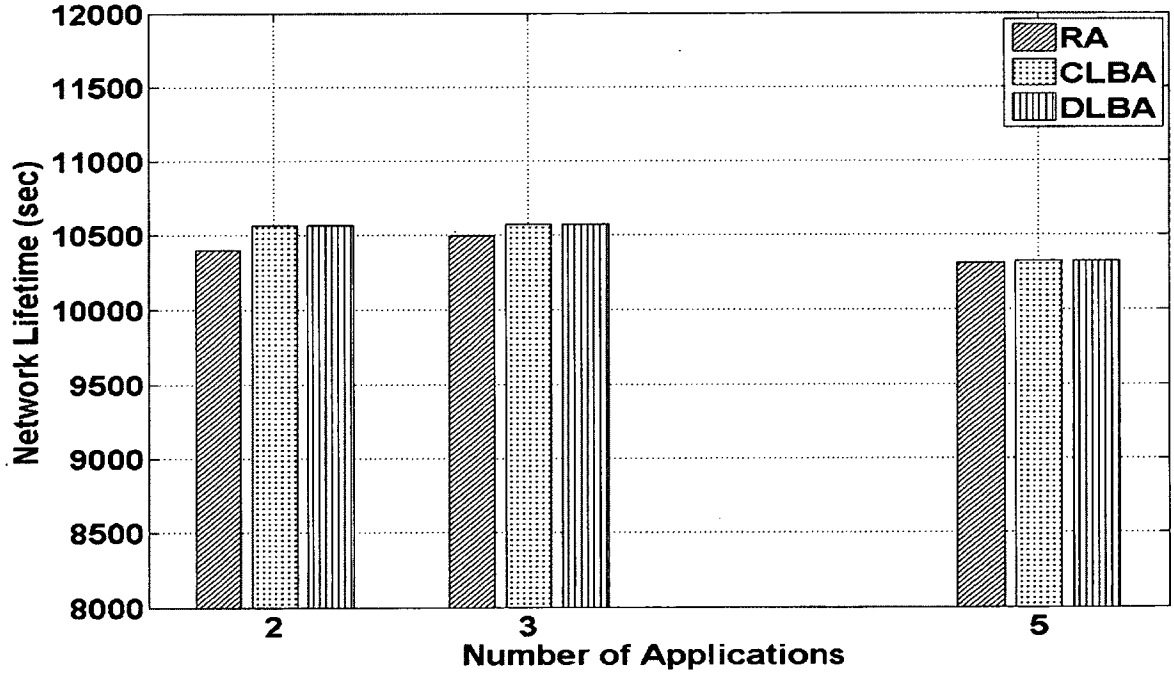
With this experiment the number of applications hosted by the WSN is varied. For a given arrival rate, the number of applications hosted by the WSN is increased from 2 to 3, and 5.

5.5.1 Effect of Number of Applications on the Performance of Static Allocation Algorithms

The effect of number of applications on the performance of static allocation algorithms is shown in Figure 5-15. When the number of applications is either 2 or 3, the performance of RA is inferior to the CLBA algorithm and the DLBA algorithm. As each application requires only 5 sensor nodes at each hop out of the available 25 nodes, for the CLBA algorithm or the DLBA algorithm only 10 nodes or 15 nodes are allocated for the two application case and the three application case respectively. For the CLBA algorithm and the DLBA algorithm, the nodes with higher energy consumption are not allocated (see Section 5.1.1). For the RA algorithm all nodes have an equal probability of allocation even for the 2 application case or the 3 application case. With RA, the sensor node with high energy consumption may also be allocated increasing the rate of energy drain for that node even further. Therefore, when the number of applications is 2 or 3, RA demonstrates a slightly inferior performance than CLBA or DLBA. When the number of applications is increased to 5, CLBA and DLBA also allocate all the available nodes and perform similar to RA (see Figure 5-15). The difference in the performance of static allocation algorithms decreases with the increase in the number of applications.



(a)



(b)

Figure 5-15: Effect of Number of Applications on the Performance of Static Allocation Algorithms

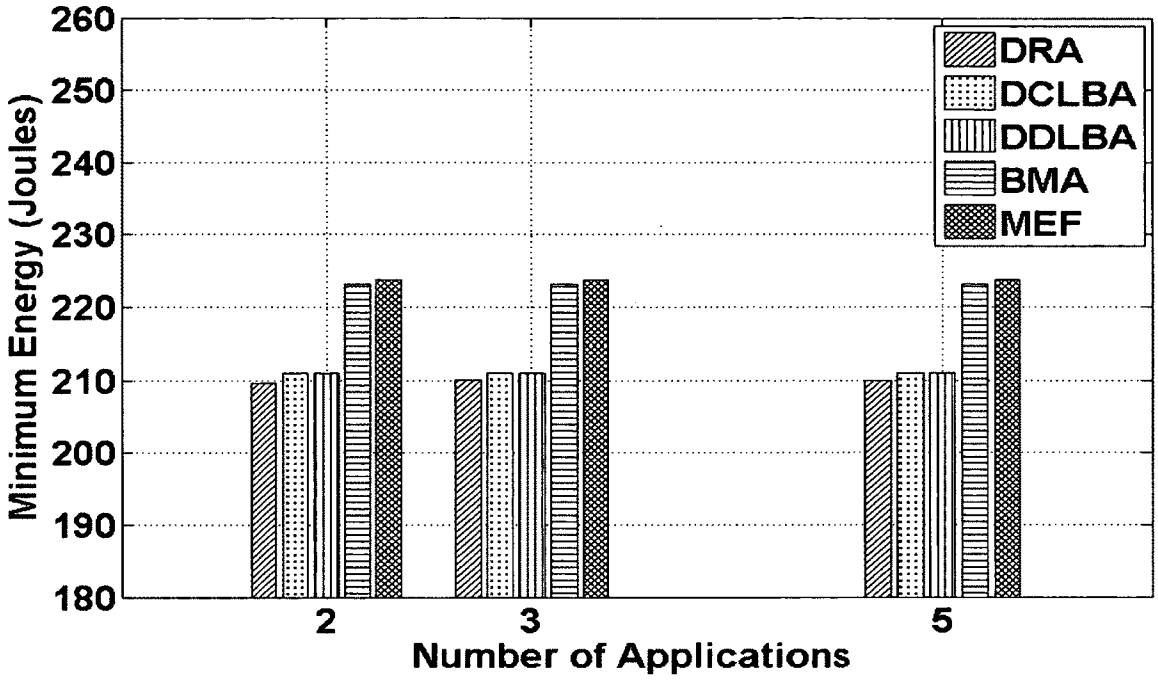
(a) Minimum Energy at a Sensor Node (b) Network Lifetime

5.5.2 Effect of Number of Applications on the Performance of Dynamic Allocation Algorithms

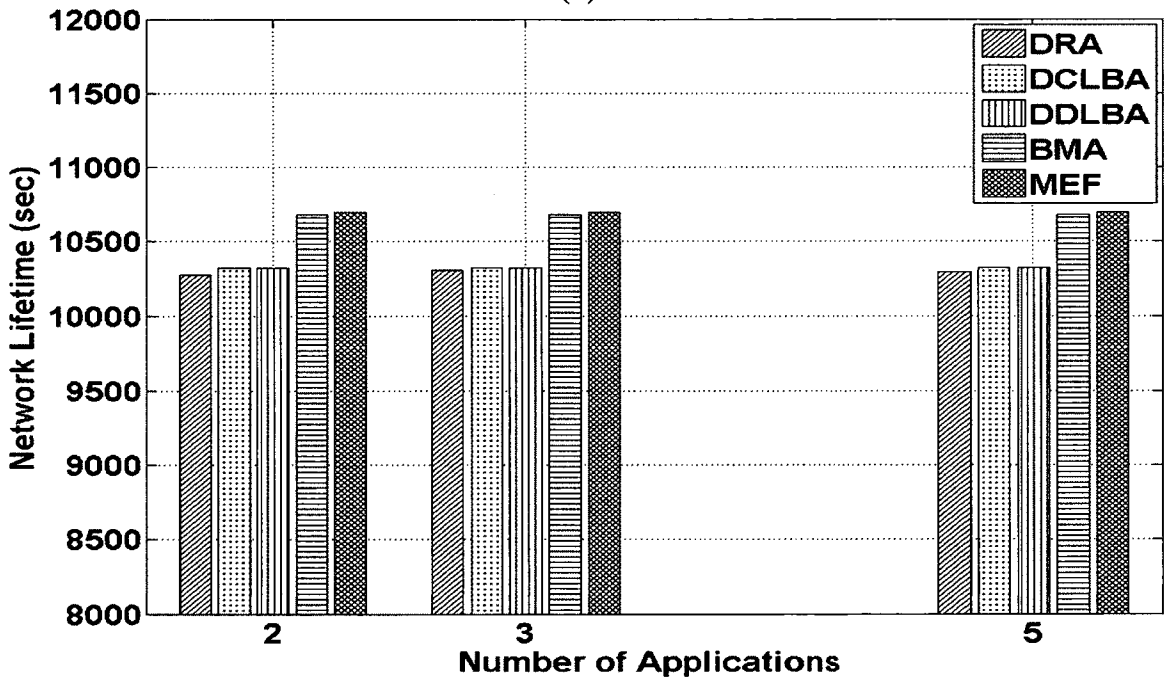
The performance of the dynamic allocation algorithms is shown in Figure 5-16. Similar to the previous experiments, the BMA algorithm and the MEF algorithms demonstrate the best performance. DRA, DCLBA, and DDLBA perform inferior to BMA and MEF. As discussed earlier, DRA allocates the sensor nodes at random and DCLBA or DDLBA balance only partial components of energy consumption. Therefore, DRA, DCLBA, and DDLBA may also allocate sensor nodes which have a higher value of total energy consumption and lower value of current available energy. The difference in the performance in the performance of the algorithms remains the same with the increase in the number of applications as the applications have similar characteristics and the arrival rate is equally split amongst the applications.

5.5.3 Comparison of Static and Dynamic Allocation Algorithms

BMA and MEF demonstrate the best performance amongst both the static and the dynamic allocation algorithms. CLBA and DLBA demonstrate the best performance amongst static allocation algorithms. The performance comparison of CLBA and BMA is shown in Figure 5-17. The difference in performance of CLBA and BMA increases as the number of applications is increased. As the number of applications is increased, the probability of allocation of sensor nodes with high energy consumption increases for the static allocation algorithms. This is because with the increase in number of applications more nodes are allocated in case of static allocation algorithms as compared to the number of nodes allocated with lesser number of applications.



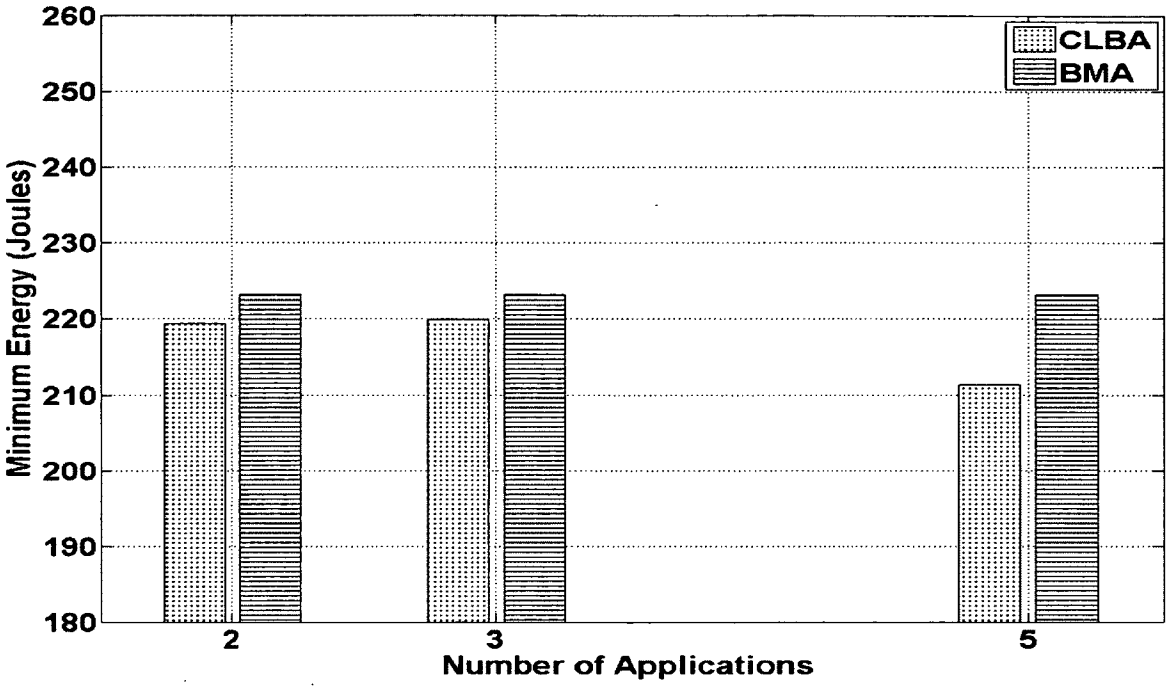
(a)



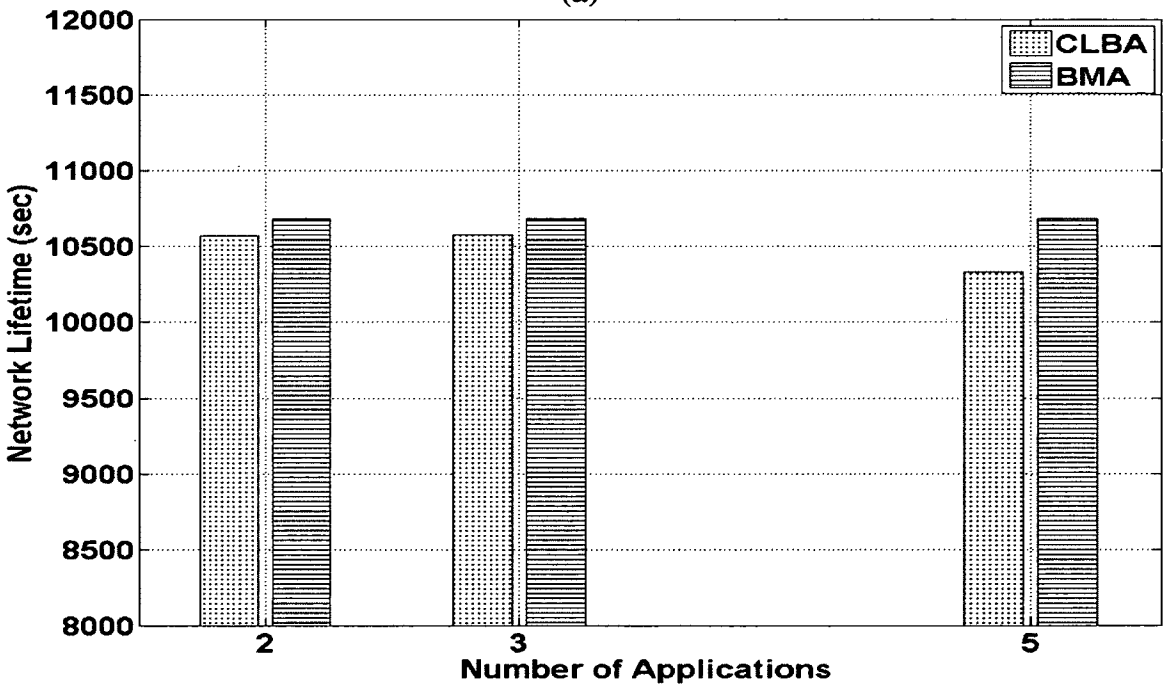
(b)

Figure 5-16: Effect of Number of Applications on the Performance of Dynamic Allocation Algorithms

(a) Minimum Energy at a Sensor Node (b) Network Lifetime



(a)



(b)

Figure 5-17: Performance Comparison of Static and Dynamic Allocation for Varying Number of Applications

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

5.6 Effect of Unequal Initial Energy of Sensor Nodes on the Performance of Allocation Algorithms

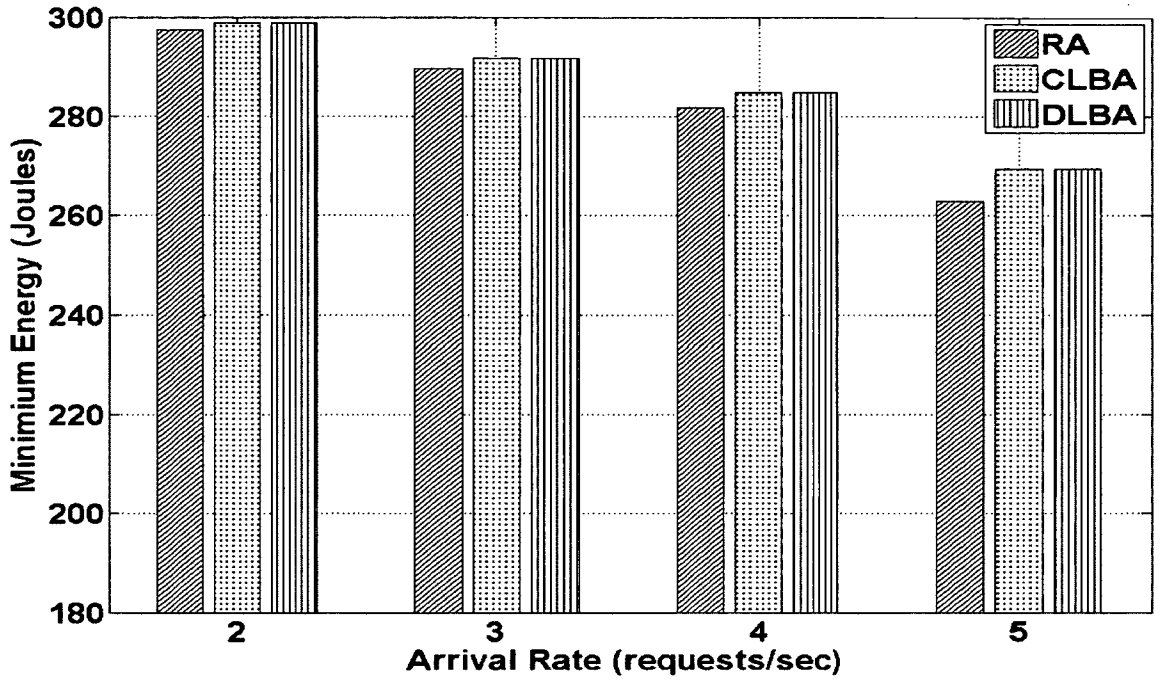
In the previous section, the performance of allocation algorithms was studied for sensor nodes with same value of initial energy. All sensor nodes had initial energy of 1000 Joules. In certain environments, initial energy of sensors may become skewed after the sensors have been in operation for a certain period of time. To investigate such an environment, in the set of experiments presented in this section, the sensor nodes are assumed to have an initial energy that is uniformly distributed between 750 Joules and 1000 Joules.

5.6.1 Effect of Arrival Rate on the Performance of Allocation Algorithms for Unequal Initial Energy of Sensor Nodes

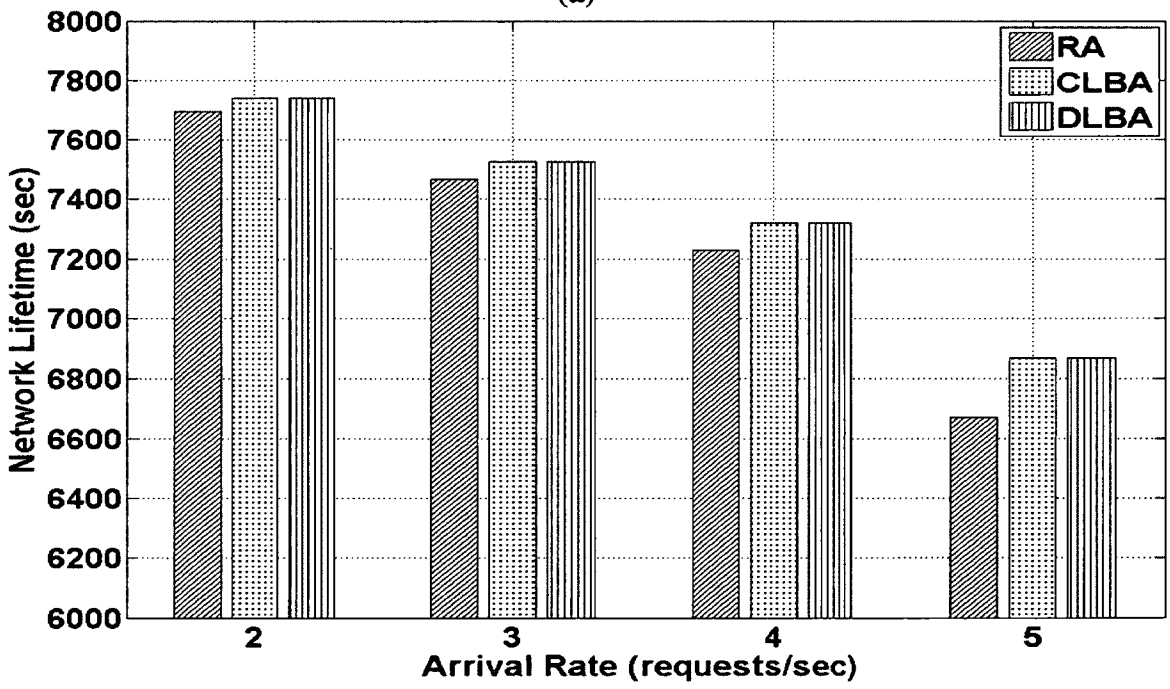
In this experiment, the arrival rate is varied. All other system and workload parameters are held at the default values (see Table 5-1 and Table 5-2).

5.6.1.1 Performance of Static Allocation Algorithms

The performance of the static allocation algorithms is shown in Figure 5-18. Similar to the observations in the experiment with sensor nodes having equal amount of initial energy (see Section 5.1.1), the performance of the CLBA and the DLBA algorithms is slightly superior to that of the RA algorithm. Both the applications require 5 sensors out of the 25 available sensors at each hop. As only 10 sensor nodes are allocated out of the available 25, the CLBA and DLBA algorithms do not allocate the sensor node with the highest energy consumption.



(a)



(b)

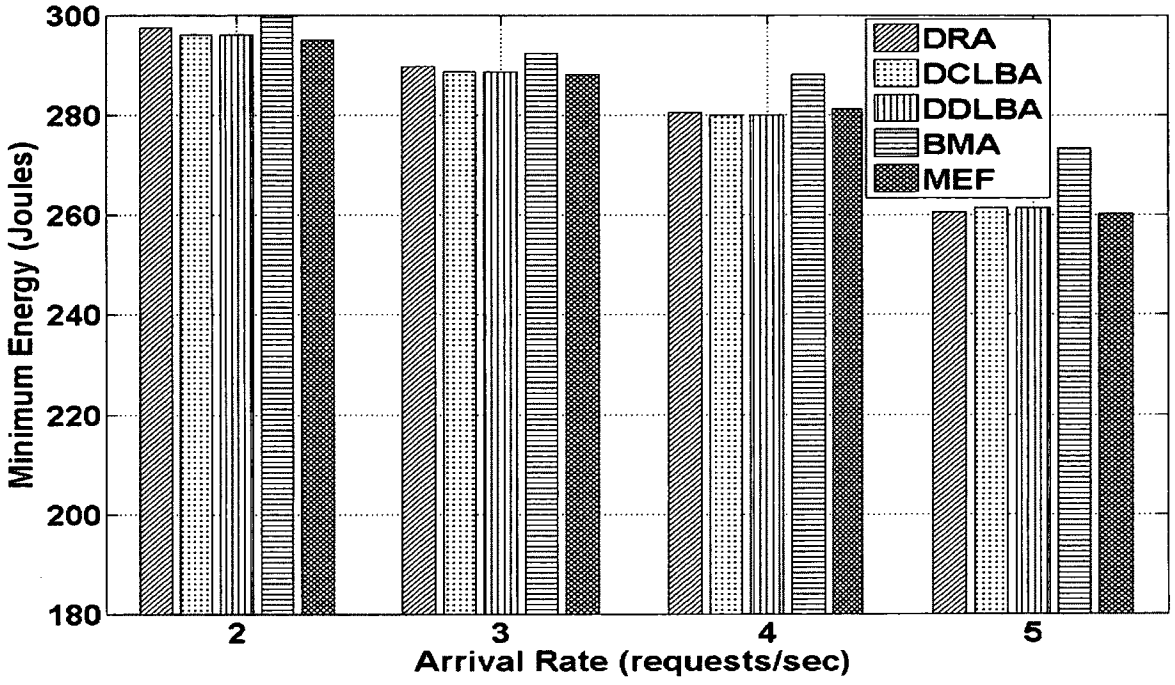
Figure 5-18: Effect of Arrival Rate on the Performance of Static Allocation Algorithms for Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

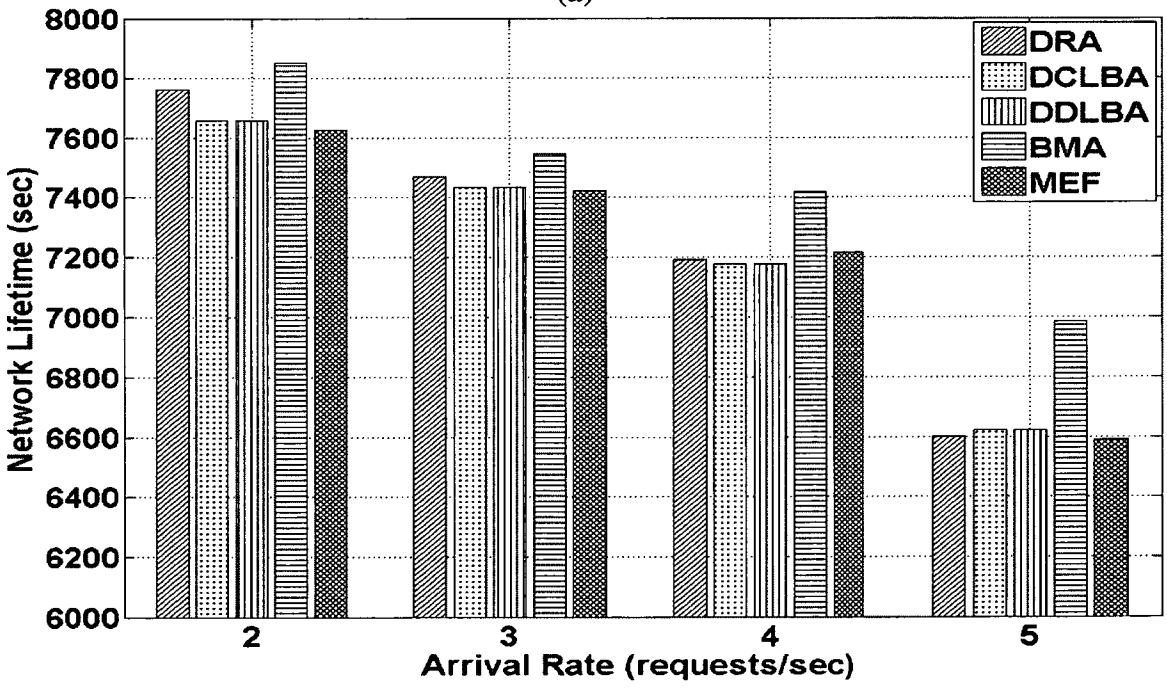
With the RA algorithm, each sensor node has an equal probability of allocation and therefore the sensor node with the highest energy consumption is also allocated. Also, with the RA algorithm, a sensor node may be allocated to requests from both the applications. This causes a more non-uniform allocation of sensor nodes with RA as compared to CLBA and DLBA. At a high arrival rate, the RA algorithm also demonstrates a smaller network lifetime (see Figure 5-18 (b)).

5.6.1.2 Performance of Dynamic Allocation Algorithms

The performance of the dynamic allocation algorithms for a WSN with sensor nodes having unequal amount of initial energy is shown in Figure 5-19. The BMA algorithm demonstrates the best performance. Unlike the WSNs in which all the sensor nodes have the same amount of initial energy, the MEF algorithm demonstrates an inferior performance as compared to the BMA algorithm. This may be attributed to the non- uniform allocation of sensor nodes with the MEF algorithm. The sensor nodes with higher amount of energy are allocated higher number of times as compared to sensor nodes with lesser amount of energy. The number of times the sensor nodes located within 1 hop distance from the cluster head get allocated for the various dynamic allocation algorithms during the entire simulation run is given in Table 5-10. From Table 5-10, it can be observed that for the MEF algorithm, only sensor nodes with id = 3, 7, 10, 16, 23, and 24 are allocated to requests for the applications. The other sensor nodes at 1 hop distance remain unallocated. This causes non-uniform distribution of load amongst the sensor nodes that in turn leads to an inferior performance of MEF in comparison to BMA.



(a)



(b)

Figure 5-19: Effect of Arrival Rate on the Performance of Dynamic Allocation Algorithms for Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

Table 5-10: Total Number of Allocations for Sensor Nodes at 1 Hop Distance for Dynamic Allocation Algorithms at Arrival Rate = 2 requests/second

Algorithm / Node	DRA	DCLBA	BMA	MEF
1	3987	4003	7122	0
2	3973	4003	7219	0
3	3962	4003	7231	12435
4	4000	4003	7242	0
5	4046	4003	7041	0
6	3939	4004	2996	0
7	4033	4004	2697	20019
8	3990	4004	2631	0
9	3938	4004	2664	0
10	3997	4004	3004	7584
11	3969	4004	3017	0
12	3939	4004	2985	0
13	4042	4004	3040	0
14	4029	4004	3042	0
15	3983	4004	3074	0
16	4115	4004	3121	20019
17	4011	4004	3094	0
18	4025	4004	3088	0
19	4034	4004	3133	0
20	4015	4004	3544	0
21	3887	4004	3546	0
22	4024	4004	3533	0
23	4021	4004	4019	20019
24	4046	4004	8012	20019
25	4090	4004	0	0

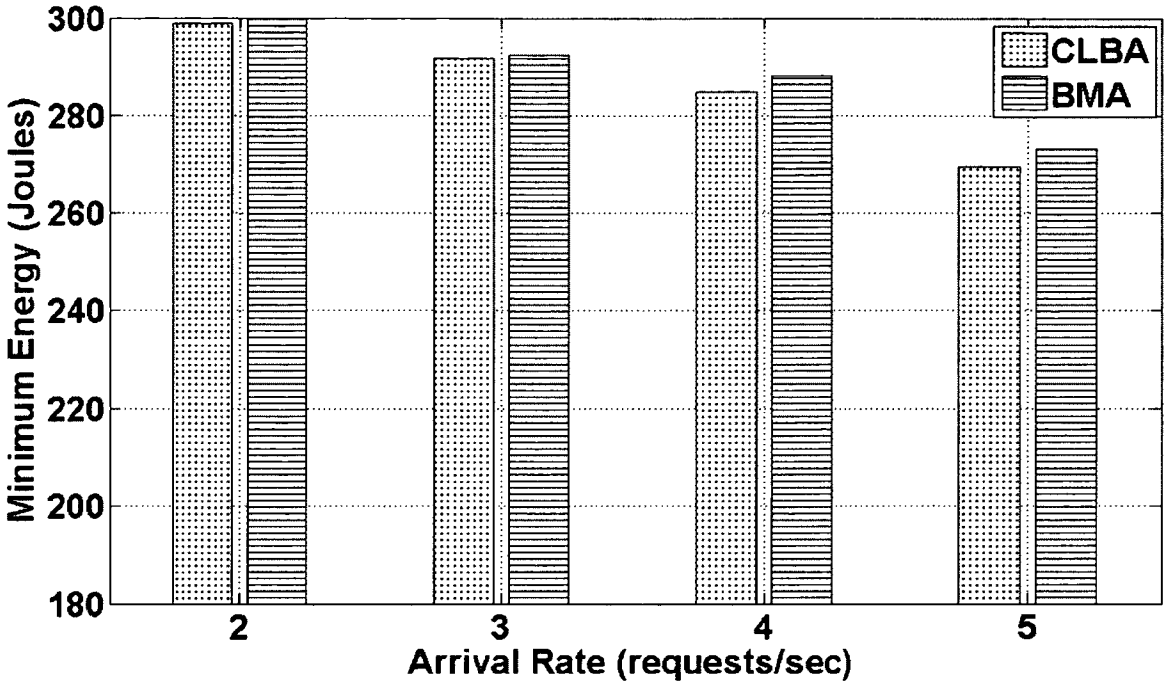
5.6.1.3 Comparison of Static and Dynamic Allocation Algorithms

Amongst both the static and dynamic allocation algorithms, the BMA algorithm demonstrates the best performance. The BMA algorithm takes an allocation decision based on the total energy consumption of sensor nodes. Therefore, it produces a more uniform distribution of load amongst the sensor nodes. The performance of CLBA and DLBA is slightly inferior to BMA. The performance comparison of CLBA and BMA is presented in Figure 5-20. As explained earlier in Section 5.1, for the default case, the number of sensors required by the applications is much less than the number of available sensors. For CLBA, the node that has consumed a high amount of energy so far is not allocated.

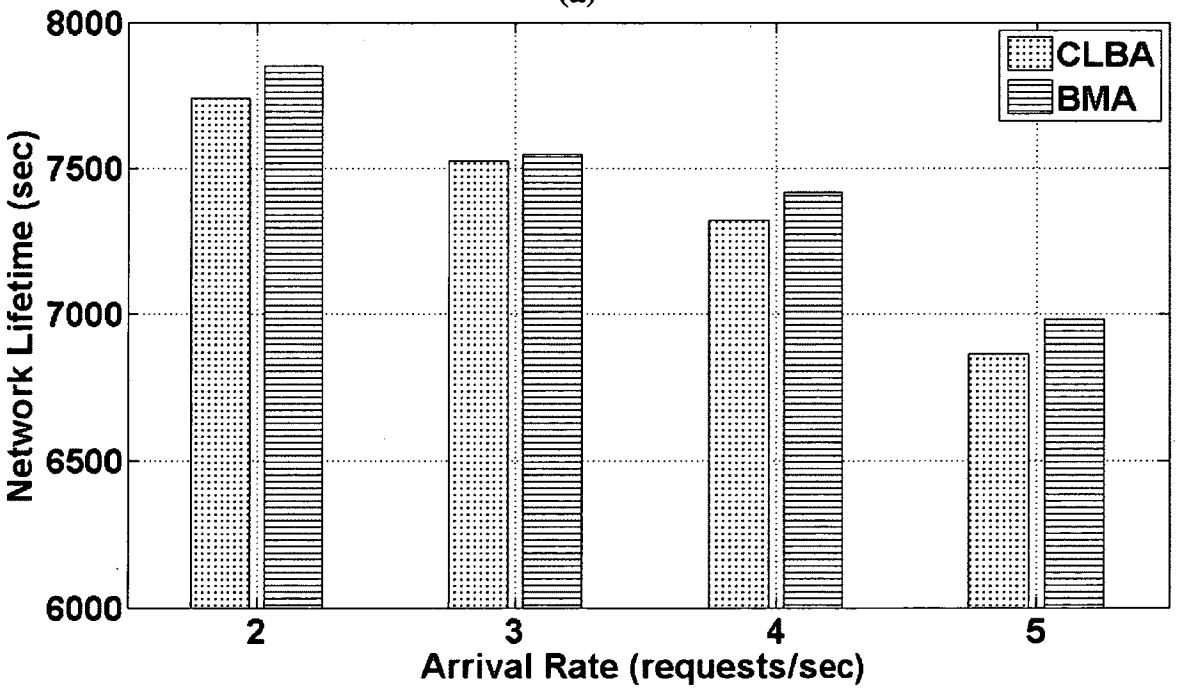
5.6.2 Effect of Other Parameters on the Performance of Allocation Algorithms for Unequal Initial Energy of Sensor Nodes

The effects of other parameters on performance were investigated. In each experiment, BMA demonstrated the best performance. As discussed in Section 5.6.1.2, the performance of MEF was observed to be inferior to that of BMA.

The impact of execution time, data size, number of sensors, and number of applications was similar to that discussed in Section 5.2, Section 5.3, Section 5.4, and Section 5.5 respectively. Thus, no further discussion is provided. The graphs from these experiments are however presented in Appendix C: Effect of Execution Time (C.1 and C.2), Effect of Data Size (C.3 and C.4), Effect of Number of Sensors (C.5 and C.6), and Effect of Number of Applications (C.7 and C.8).



(a)



(b)

Figure 5-20: Performance Comparison of Static and Dynamic Algorithms for Varying Arrival Rate for Unequal Initial Energy of Sensor Nodes

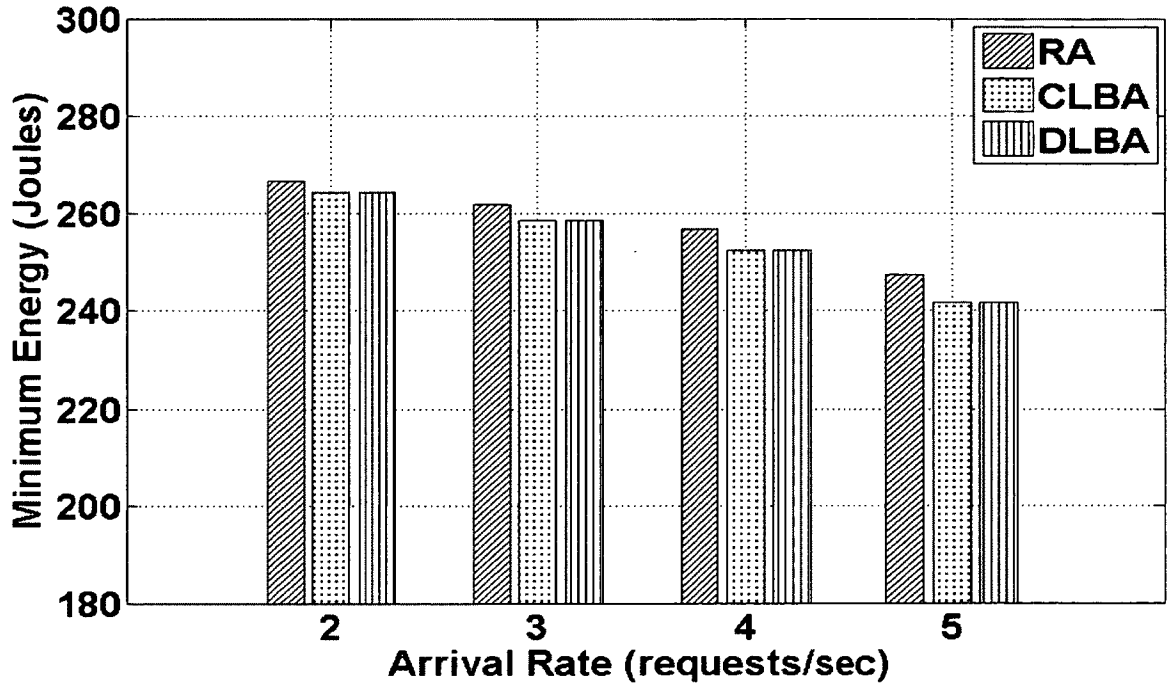
(a) Minimum Energy at a Sensor Node (b) Network Lifetime

5.7 Performance of Allocation Algorithms for Applications with Greater Variation

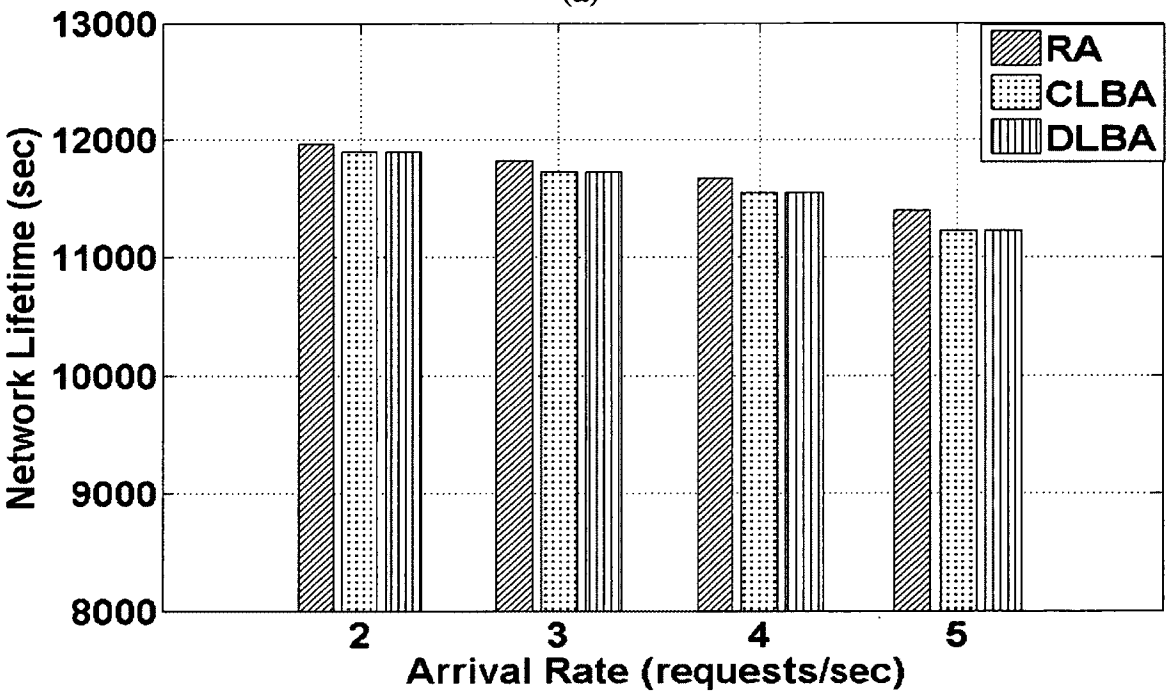
For most of the experiments discussed so far in this chapter, 2 applications are considered. The applications have similar requirements in terms of the total number of sensors required and the number of sensors required at the various hop distances from the cluster head. In this experiment, 5 applications are considered. The set of applications is similar to the default set of applications considered while evaluating the performance of scheduling algorithms (see Table 4-3). The data size and execution time of applications and the initial energy of the sensor nodes are all held at their default values (see Table 5-2, and Table 5-3). Note that the applications vary in the total number of sensors required by the applications and also the number of sensors required by the various applications at each hop distance.

5.7.1 Performance of Static Allocation Algorithms

The arrival rate is varied. The performance of the static allocation algorithms is shown in Figure 5-21. The CLBA and the DLBA algorithms demonstrate a comparable performance similar to the other experiments presented in this chapter. The RA algorithm performs slightly better than the CLBA and the DLBA algorithms. For the default case of 2 applications, the number of sensors required by the applications at each hop is much less than the number of available sensors. For such a case, the sensor nodes with higher energy consumption were not allocated for CLBA and DLBA (see Section 5.1.1).



(a)



(b)

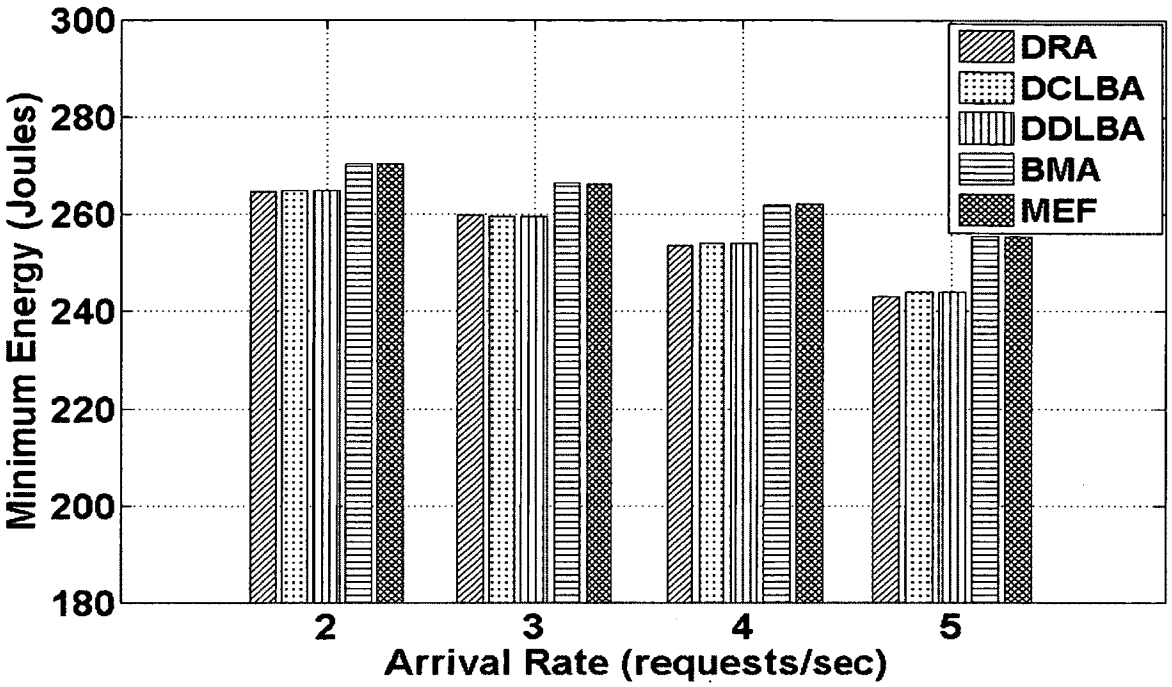
Figure 5-21: Performance of Static Allocation Algorithms for Applications with Greater Variation

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

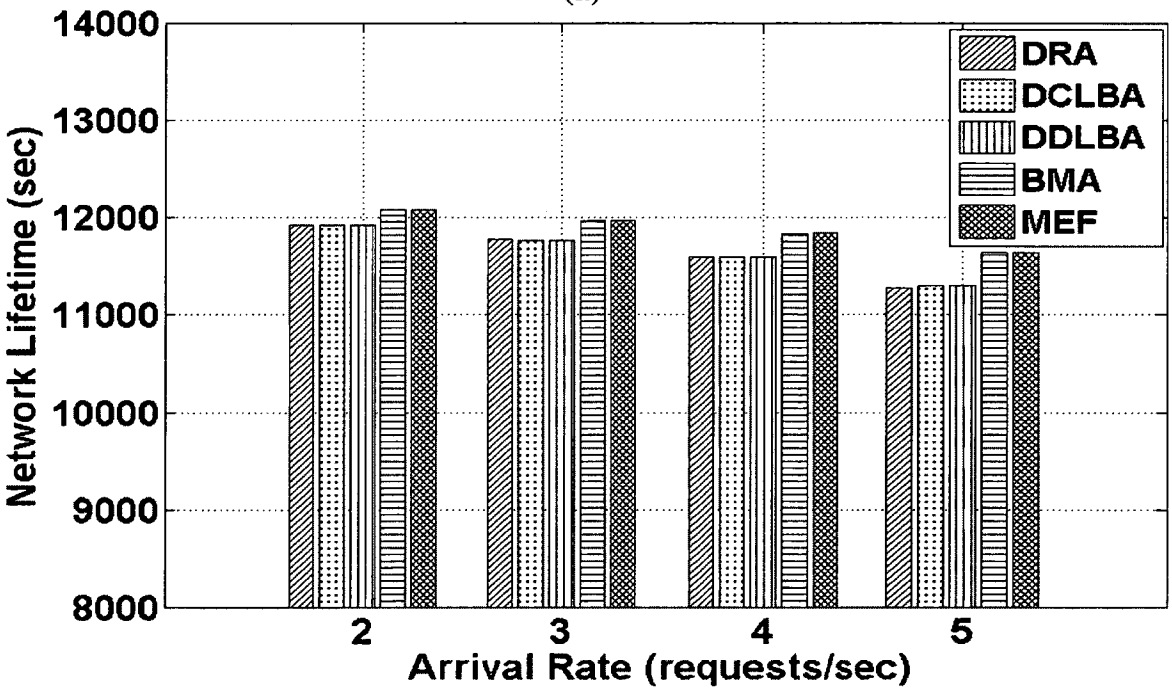
In this experiment, some of the applications require a higher number of sensors at the various hop distances. In case of RA, the nodes with higher value of total energy consumption may or may not be allocated as all nodes have an equal probability of selection. In case of CLBA and DLBA, these nodes are allocated as CLBA and DLBA focus on balancing the energy consumption amongst the CPU component or a partial component of energy consumption at the radio component of the sensor nodes respectively. The CLBA and the DLBA algorithms also demonstrate a slightly smaller value of network lifetime as compared to the RA algorithm (see Figure 5-21 (b)).

5.7.2 Performance of Dynamic Allocation Algorithms

The performance of the dynamic allocation algorithms as the arrival rate is varied is shown in Figure 5-22. The relative performances of the dynamic allocation algorithms are similar to that observed in the experiment with default set of applications presented earlier in this chapter (see Section 5.1.2). The BMA algorithm and the MEF algorithm demonstrate the best performance. As explained in Section 5.1.2, the DRA algorithm allocates sensor nodes to requests for application at random without considering any characteristics of the applications or the network. The DCLBA algorithm and the DDLBA algorithm allocate sensor nodes to requests for applications based on the knowledge of partial energy consumption at the sensor nodes and hence also allocate sensor nodes with greater energy consumption or lower value of available energy to requests. Therefore, DRA, DCLBA, and DDLBA demonstrate an inferior performance as compared to BMA and MEF.



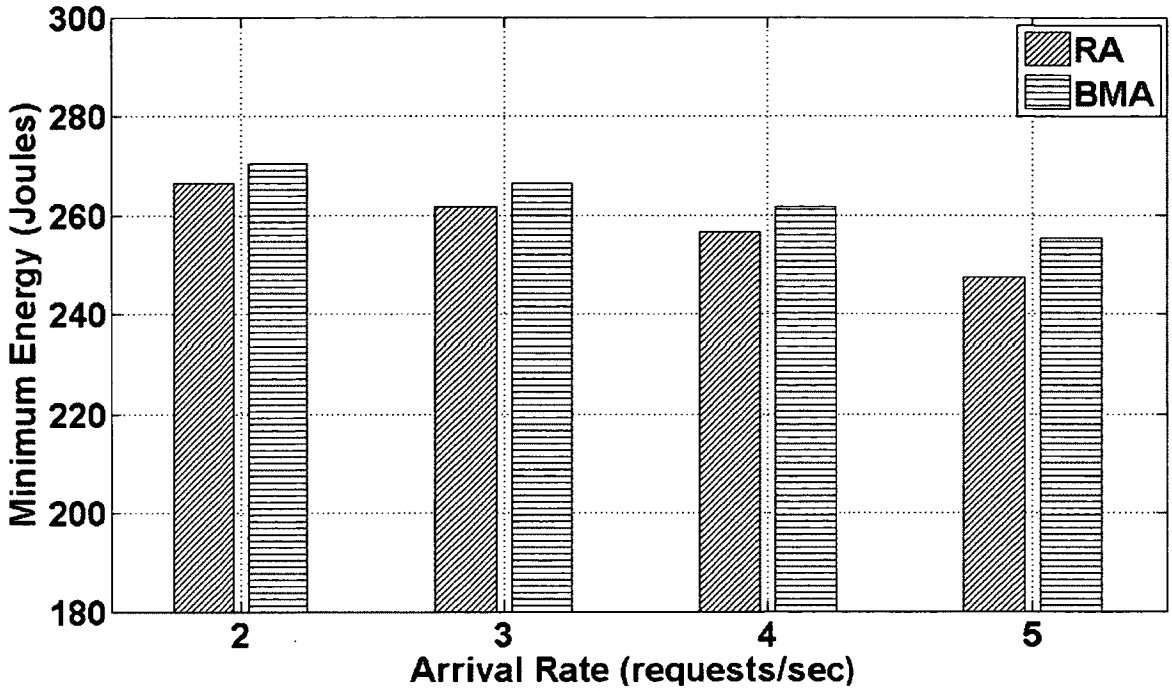
(a)



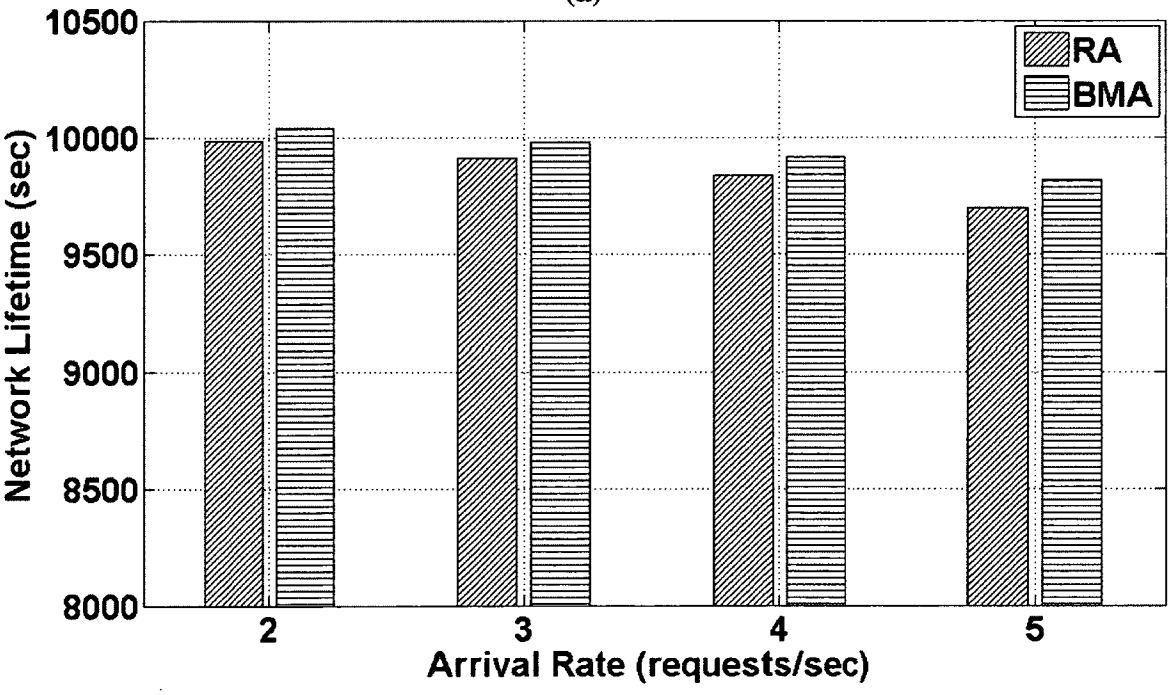
(b)

Figure 5-22: Performance of Dynamic Allocation Algorithms for Applications with Greater Variation

(a) Minimum Energy at a Sensor Node (b) Network Lifetime



(a)



(b)

Figure 5-23: Performance Comparison of Static and Dynamic Allocation for Applications with Greater Variation

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

5.7.3 Comparison of Static and Dynamic Allocation Algorithms

Amongst dynamic allocation algorithms, BMA and MEF demonstrate the best performance. BMA and MEF perform dynamic allocation and efficiently balance the energy amongst the sensor nodes comprising the WSN. Amongst the static allocation algorithms, RA demonstrated a slightly superior performance as compare to CLBA and DLBA. The performance comparison of RA and BMA, which are the best or one of the best performing algorithms in the category of static and dynamic allocation respectively is shown in Figure 5-23. Similar to the observations made in the earlier experiments presented in this chapter, BMA once again performs the best performance amongst both the static and dynamic algorithms.

5.8 Performance Analysis of the Proposed Allocation Algorithms

An experiment has been performed to compare the performance of BMA to the best performance that can be achieved with a static allocation. In order to limit the number of possible allocations a 2 hop WSN with 5 sensors at each hop is considered. The WSN hosts two applications. Each application requires a total of 4 sensors (2 sensors at each hop). The rest of the parameters are held at the default values (see Table 5-1 and Table 5-2). As the parameter values for Application 1 and Application 2 are the same, the static allocation of sensor nodes to either of the application is interchangeable. Also, in order to balance the load amongst the sensor nodes, none of the sensor nodes is allocated to both the applications. As each application requires 2 sensor nodes at each hop distance, for every static allocation, there is one unallocated sensor node out of the available five sensor nodes at each hop distance. For each unallocated node at one hop distance, there

are 5 possibilities of an unallocated node at a distance of 2 hops. This yields a total of 25 different allocation possibilities. Table 5-11 lists all the possible allocations. Allocation for Application 1 and Application 2 are represented by A1 and A2 in column two and three of Table 5-11 respectively. The allocation defines the set of sensors that are allocated to a particular application. $S_{i,j}$ represents a sensor node where i denotes the hop distance from the cluster head and j denotes the id of the sensor. The id of a sensor at each hop is a number from 1 to 5 as there are five sensors at each hop.

All possible 25 static allocations are experimented with. The performance of the BMA algorithm is compared to all the possible allocations, as the BMA algorithm is the best performing algorithm amongst the algorithms proposed in this research. The minimum energy level and network lifetime are observed at the end of the simulation period for the 25 possible static allocations and also for the BMA algorithm (see Table 5-11).

The minimum energy level observed with the BMA algorithm is only 7.6% lower than the minimum energy level observed with the best performing allocation. The difference in the network lifetime observed with the BMA algorithm and the best performing allocation is only 3.7%. This demonstrates that by balancing the total energy consumption amongst the sensor nodes, performance close to the best possible performance can be achieved.

Table 5-11: Performance Analysis of BMA

Allocation	A1	A2	Minimum Energy (Joules)	Network Lifetime (sec)
BMA			241.1973107	11569.33135
1	{(S12,S13),(S22,S23)}	{(S14,S15),(S24,S25)}	208.9133403	10932.75351
2	{(S12,S13),(S21,S23)}	{(S14,S15),(S24,S25)}	208.911066	10932.75409
3	{(S12,S13),(S21,S22)}	{(S14,S15),(S24,S25)}	209.8502832	11431.28814
4	{(S12,S13),(S21,S22)}	{(S14,S15),(S23,S25)}	208.9179329	10932.75399
5	{(S12,S13),(S21,S22)}	{(S14,S15),(S23,S24)}	234.3633643	11429.50195
6	{(S11,S13),(S22,S23)}	{(S14,S15),(S24,S25)}	208.9117108	10932.7537
7	{(S11,S13),(S21,S23)}	{(S14,S15),(S24,S25)}	208.9124394	10932.75351
8	{(S11,S13),(S21,S22)}	{(S14,S15),(S24,S25)}	208.9179329	10932.75347
9	{(S11,S13),(S21,S22)}	{(S14,S15),(S23,S25)}	208.9116666	10932.75325
10	{(S11,S13),(S21,S22)}	{(S14,S15),(S23,S24)}	234.3575374	11429.50206
11	{(S11,S12),(S22,S23)}	{(S14,S15),(S24,S25)}	208.9130731	10932.75395
12	{(S11,S12),(S21,S23)}	{(S14,S15),(S24,S25)}	208.9179329	10932.75371
13	{(S11,S12),(S21,S22)}	{(S14,S15),(S24,S25)}	208.9115783	10932.75379
14	{(S11,S12),(S21,S22)}	{(S14,S15),(S23,S25)}	208.9149212	10932.75332
15	{(S11,S12),(S21,S22)}	{(S14,S15),(S23,S24)}	234.3589439	11429.50167
16	{(S11,S12),(S22,S23)}	{(S13,S15),(S24,S25)}	208.9132917	10932.75367
17	{(S11,S12),(S21,S23)}	{(S13,S15),(S24,S25)}	208.9132917	10932.75366
18	{(S11,S12),(S21,S22)}	{(S13,S15),(S24,S25)}	208.9147225	10932.75369
19	{(S11,S12),(S21,S22)}	{(S13,S15),(S23,S25)}	208.9179329	10932.75355
20	{(S11,S12),(S21,S22)}	{(S13,S15),(S23,S24)}	234.3588953	11429.50176
21	{(S11,S12),(S22,S23)}	{(S13,S14),(S24,S25)}	236.1515573	11466.25413
22	{(S11,S12),(S21,S23)}	{(S13,S14),(S24,S25)}	236.1499323	11466.25372
23	{(S11,S12),(S21,S22)}	{(S13,S14),(S24,S25)}	236.1561544	11466.25377
24	{(S11,S12),(S21,S22)}	{(S13,S14),(S23,S25)}	236.151469	11466.25332
25	{(S11,S12),(S21,S22)}	{(S13,S14),(S23,S24)}	261.6015858	12013.33256

5.9 Discussion of Simulation Results

This section provides a brief discussion of the simulation results to gain insights into the performance of the allocation algorithms.

Experiments have been performed by varying the arrival rate of incoming application requests, execution time of applications, data size of request and response messages, number of sensors required by applications, and the number of applications. Experiments have been performed for sensor nodes with equal amount of initial energy and also for sensor nodes with unequal amount of initial energy. The key findings from the experiments are summarized next.

- The BMA algorithm and the MEF algorithm that perform dynamic allocation based on the total energy consumption of the sensor nodes or the current value of energy at the sensor nodes demonstrate the best performance. The other dynamic allocation algorithms DRA, DCLBA, and DDLBA demonstrate an inferior performance as compared to BMA and MEF as they allocate sensor nodes at random or based on knowledge of partial energy consumption at sensor nodes.
- For higher values of execution time of Application 1, the static allocation algorithms demonstrate a much inferior performance as compared to the dynamic allocation algorithms. For static allocation, the sensor nodes that get allocated to Application 1 have very high energy consumption at the CPU component for higher values of execution time of Application 1. The RA algorithm demonstrates the worst performance because RA leads to a higher non-uniformity in the distribution of energy consumed as compared to CLBA and DLBA. Amongst the dynamic allocation algorithms, the BMA and the MEF algorithms demonstrate the best performance.

- The minimum energy at a sensor node at the end of the simulation period decreases for all the algorithms with increase in the size of the messages. Also, the network lifetime decreases for all the algorithms and the rate of energy drain increases for all algorithms with an increase in the size of messages. This may be attributed to the fact that the energy consumption during transmission of a message increases with the increase in the size of the message. The BMA algorithm and the MEF algorithm demonstrate the best performance as they do not allocate requests to sensor nodes that have higher value of energy consumption or lower value of current energy. The DRA algorithm allocates requests at random. Also, the DCLBA and the DDLBA algorithms aim to balance only some of the components of energy consumption amongst the sensor nodes and hence allocate sensor nodes irrespective of their total energy consumption or the current level of energy at the sensor nodes. As a result, they demonstrate an inferior performance in comparison to the MEF and the BMA algorithms.
- The performance of the algorithms was studied for varying number of sensors required by the applications. The BMA algorithm and the MEF algorithm demonstrate the best performance in all the cases. The CLBA algorithm and the DLBA algorithm provide a performance comparable to the BMA algorithm when the numbers of sensors required by the applications are much less than the total number of available sensors. This is because the sensor nodes with high energy consumption may not get allocated for these cases. When the number of sensors required by the applications is increased, the sensor nodes with high energy consumption are

allocated for CLBA and DLBA. Hence their performance is inferior to the BMA algorithm and the MEF algorithm.

- As the number of applications hosted by the WSN is increased, the difference in performance of the static allocation algorithms as compared to the BMA algorithm or the MEF algorithm increases. As the number of applications is increased, the probability of allocation of sensor nodes with high energy consumption increases for the static allocation algorithms.
- Experiments have also been performed for a higher number of applications hosted by the WSN with greater variations in the total number of sensors required by the applications and the number of sensors required by the applications at each hop, with other parameters held at the default values. BMA and MEF, once again demonstrate the best performance.
- Experiments have been performed for WSNs comprising of sensor nodes with unequal amount of initial energy. Unlike the wireless sensor networks in which all the sensor nodes have the same amount of initial energy, the MEF algorithm demonstrates an inferior performance as compared to the BMA algorithm.

Chapter 6: Conclusions

This chapter summarizes the important contributions of this research. Multi-purpose wireless sensor networks that provide an effective sharing of resources among multiple applications are gaining in popularity. This research has focused on such WSNs hosting multiple applications. When a wireless sensor network serves more than one application, resource management of sensor nodes becomes even more challenging. Resource management in a WSN hosting multiple applications is a two-step process. Scheduling determines the order in which the requests for various applications that are submitted by the users of the applications are executed. Allocation determines the sensor nodes that are to be selected from a set of sensor nodes to execute the job requests corresponding to an application request. The main aim of any scheduling algorithm is to improve the overall mean response time to the users of the applications. The main aim of an allocation algorithm is to improve the lifetime of the WSN.

In this thesis a number of scheduling and allocation algorithms are proposed. The important conclusions of this research on scheduling are summarized in Section 6.1 and the important conclusions of this research on allocation are summarized in Section 6.2.

6.1 Scheduling Algorithms

The important contributions of this thesis include five new algorithms in the field of scheduling multiple applications in a shared WSN. Important attributes of each algorithm that was discussed in more detail in Chapter 3 and Chapter 4 are presented.

- LNSF and LNHF make scheduling decisions based only on the information of application or network respectively.
- LNDPF uses information about both the application and the network. It considers all sensors required by an application while making a scheduling decision. With this algorithm, the application requests are scheduled in non-decreasing order of the Number Distance Product of applications.
- LFNDPF uses information about sensors that are located farthest away from the cluster head. This algorithm schedules applications in non-decreasing order of their Farthest Number Distance Product.
- LWFNDPF computes the weighted FNDP for the applications. With the LWFNDPF algorithm, applications are scheduled in non-decreasing order of WFNDP for the applications. Higher hop distances are assigned a higher weight value.

Simulation experiments are performed to get insights into the performance of the proposed algorithms. Experiments have been performed to study the effect of arrival rate of incoming requests, data size of request and response messages, allocation of sensors, deployment of sensors, number of applications, and probability of arrival of requests from an application. Effect of the underlying MAC layer protocol has also been studied. The key observations from these experiments are summarized next.

- Knowledge of both network and application is important in making a scheduling decision.
- Communication delays play a very significant role in WSNs. The messages to and from the sensors that are located farthest away from the cluster head experience maximum delays. The greater the distance of a sensor from the cluster head in terms

of number of hops, greater are the delays experienced by the messages to and from the sensor.

- The Least Weighted Farthest Number Distance Product First algorithm provides the best performance for most of the configurations experimented with.
    - Amongst the sensors required by an application, LWFNDPF considers the sensors located farthest away from the cluster head.
    - The farthest sensors required by an application play an important role in determining the response time of an application and hence the overall mean response time. This is because response for an application request is complete only when the response for all job requests corresponding to the application request is received back by the proxy.
    - Delays are experienced at each hop due to interference from the nearby sensors. The superior performance of LWFNDPF over LFNDPF is attributed to its allocating a higher weight to higher hop distances.
- The difference in the performance of the algorithms increases with the increase in the arrival rate of incoming requests, and data size of messages as each of these correspond to increase in delays experienced by messages in the network.
- A more detailed performance analysis of the LWFNDPF algorithm is done. All possible orders/schedules are experimented with in a three application case and a four application case. The objective of this analysis is to see how close the LWFNDPF algorithm performs to the best performer. For the three application case, the schedule corresponding to the LWFNDPF algorithm provides the best performance. For the four application case, at a high arrival rate, the performance of the LWFNDPF

algorithm is observed to be superior to that achieved with nineteen out of the possible twenty-four schedules and is within 8.5% of the best schedule for the other five schedules.

The scheduling algorithms proposed in this thesis focus on improving the overall mean response time to the users of applications supported by the multi-purpose wireless sensor networks.

In addition to scheduling this thesis presents research on allocation algorithms that aim to improve the lifetime of a wireless sensor network. The important contributions of this thesis in the field of allocation in WSNs hosting multiple applications are presented next.

6.2 Allocation Algorithms

The contributions of this thesis include new static and dynamic algorithms for allocation of sensor nodes to requests for applications in a WSN hosting multiple applications.

The key characteristics of each algorithm that was discussed in detail in Chapter 3 and Chapter 5 are summarized next.

- For static allocation algorithms, the sensor nodes are allocated to applications *a priori* and the allocation does not change for all the requests for a given application.
    - RA allocates sensor nodes at random with each sensor node having an equal probability of being selected.
    - CLBA focuses on balancing energy consumption at sensor nodes only due to the CPU component of the sensor node.

- DLBA focuses on balancing energy consumption at the radio component of the sensor nodes only due to the transmission of response messages corresponding to requests processed by a node.

- For dynamic allocation algorithms, sensor allocation is done for each request arrival. Thus, different requests for the same application may be served by different sensor nodes.
  - DRA, DCLBA and DDLBA are dynamic versions of RA, CLBA and DLBA. Each of these uses a similar logic as its static counterpart, but makes an allocation decision dynamically upon each request arrival.
  - BMA balances the total energy consumption both due to the CPU component and the radio component amongst the sensor nodes.
  - MEF allocates sensor nodes based on the current energy level of the sensor nodes and allocates sensor nodes with higher amount of energy first.

Simulation experiments are performed to understand the impact of key parameters on performance. These parameters include arrival rate of incoming application requests, execution time of applications, data size of request and response messages, number of applications, number of sensors required by the applications, and the initial level of energy of sensor nodes. Insights into the performance of these algorithms gained from the simulation results are summarized next.

- The dynamic allocation algorithms demonstrate a superior performance as compared to the static allocation algorithms for most of the configurations experimented with.

- Dynamic allocation algorithms balance the load more uniformly as compared to static allocation algorithms because for static allocation, the allocation of sensor nodes to requests for applications remains the same for every arrival of an application request. This causes non-uniform loading of sensor nodes especially in cases of applications with higher demands. For example, for higher value of size of messages, the difference in the minimum energy at a sensor node with the CLBA algorithm and the BMA algorithm is close to 13.5% (see Section 5.3).
- Static allocation algorithms allocate sensor nodes irrespective of the total energy consumed at the sensor nodes or the current energy at the sensor nodes. This information is not known *a priori*.

- Balanced Metric Allocation demonstrates the best performance for the configurations experimented with. The superior performance of BMA demonstrates the importance of using the knowledge of the aggregate of the energy consumed by the CPU component of the sensor node and the radio component of the sensor node. Using the aggregate while allocating sensor nodes to requests for applications brings about a significant improvement in performance.
  - MEF demonstrates a performance comparable to BMA for environments in which sensor nodes have equal amounts of initial energy. For wireless sensor networks in which the sensor nodes have unequal initial energy (discussed in Section 5.6), the performance of the MEF algorithm is inferior to the performance of the BMA algorithm. As discussed in Section

5.6, this is attributed to the queuing of requests at sensor nodes with higher amount of energy. For environments where sensor nodes have unequal amount of initial energy, the MEF algorithm leads to a non-uniform distribution of load amongst sensors.

  - DRA, DCLBA, and DDLBA demonstrate an inferior performance as compared to BMA. DRA allocates sensor nodes at random. DCLBA and DDLBA balance only specific components of energy consumption at the sensor nodes.

- The difference in the performance of the algorithms increases with the increase in the arrival rate of incoming requests, execution time of applications, and data size of request and response messages. Each of these corresponds to an increase in system load indicating the increasing importance of using knowledge-based allocation algorithms for systems with high loads.

6.3 Directions for Future Research

In this research, a simulation based approach has been used to study the performance of the proposed scheduling and allocation algorithms for wireless sensor networks hosting multiple applications. A few topics for future research are presented next.

- Analyzing the performance of the proposed algorithms on a real sensor network forms an interesting direction for future research.
- Analysing the performance of the proposed algorithms for various routing protocols warrants investigation.

- The effect of heterogeneous sensor nodes on the performance of the proposed algorithms is another interesting area for future research.
- A single cluster based WSN is investigated in this research. Evaluating the performance of the proposed algorithms for multiple cluster based WSNs requires further investigation.
- Adding mechanisms for detection and handling of failures of sensor nodes to the proposed algorithms forms another interesting area for future research.

References

[1] Foster I. and Kesselman C., "The Grid: Blueprint for a New Computing Infrastructure", *Morgan Kaufmann Publishers Inc*, San Francisco, CA, U.S.A, 1999.

[2] Foster I., "What is the Grid? A three-point checklist", Argonne National Laboratory & University of Chicago, 2002, *available at http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf* [Accessed: July 15, 2012].

[3] Foster I., Kesselman C., and Tuecke S., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *The International Journal of High Performance Computing Applications*, Vol. 15, No. 3, pp. 200-222, June 2001.

[4] McKnight L., Howison J., and Bradner S., "Distributed Resource Sharing by Mobile, Nomadic, and Fixed Devices", *IEEE Journal of Internet Computing*, Vol. 8, No. 4, pp. 24-31, August 2004.

[5] Phan T., Huang L., and Dulan C., "Integrating Mobile Wireless Devices into the Computational Grid", *In the Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MOBICOM'02)*, pp. 271-278, Atlanta, GA, U.S.A., September 2002.

[6] Agarwal A., Norman D., and Gupta A., "Wireless Grids: Approaches, Architectures and Technical Challenges", *MIT Sloan School of Management Working Paper 4459-04*, January 2004.

[7] McKnight L.W., Howison J., Bradner S., "Guest Editors' Introduction: Wireless Grids--Distributed Resource Sharing by Mobile, Nomadic, and Fixed Devices", *IEEE Journal of Internet Computing*, Vol. 8, No. 4, pp. 24- 31, August 2004.

[8] Ahuja S. P., Myers Jack. R., "A Survey on Wireless Grid Computing", *The Journal of Supercomputing*, Vol. 37, No. 1, pp. 3-21, July 2006.

[9] Tham, C.K, and Buyya R., "Sensor Grid: Integrating Sensor Networks and Grid Computing", *Invited Paper in CSI Communications, Special Issue on Grid Computing*, India, July 2005.

[10] Xing G., Lu C., Pless R., and Huang Q., "On Greedy Geographic Routing Algorithms in Sensing Covered Networks", *In the Proceedings of the 5th ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc '04)*, pp. 31-42, Tokyo, Japan, May 2004.

[11] Yuriyama M., Kushida T., "Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing", *In the Proceedings of the*

13th International Conference on Network-Based Information Systems (NBiS 2010), pp. 1-8, Takayama, Gifu, Japan, September 2010.

[12] Hassan M. M., Song B., and Huh E., "A Framework of Sensor-Cloud Integration Opportunities and Challenges", *In the Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication (ICUIMC 2009)*, pp. 618-626, Suwon, Korea, January 2009.

[13] Becker D., "Wi-Fi Takes Over in Homes", 2005, available at http*://www.techrepublic.com/article/wi-fi-takes-over-in-homes/5544942*, [Accessed: July 2012].

[14] Akbar A.H., Iqbal Ali A., and Kim K., "Binding Multiple Applications on Wireless Sensor Networks", *In the Proceedings of the 1st International Conference on Advances in Grid and Pervasive Computing (GPC'06)*, pp. 250-258, Taichung, Taiwan, May 2006.

[15] Roy N., Rajamani V., Julien C., "Supporting Multi-Fidelity-Aware Concurrent Applications in Dynamic Sensor Networks", *In the Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 43-49, Mannheim, Germany, March 2010.

[16] Cid del P.J., Michiels S., Joosen W., Hughes D., "Middleware for Resource Sharing in Multi-Purpose Wireless Sensor Networks", *In the Proceedings of IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA)*, pp. 1-8, Suzhou, China, November 2010.

[17] Steffan J., Fiege L., Cilia M., and Buchmann A., "Scoping in Wireless Sensor Networks", *In the Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing*, pp. 167-171, Toronto, ON, Canada, October 2004.

[18] Yu Y., Rittle L. J., Bhandari V., and LeBrun J. B., "Supporting Concurrent Applications in Wireless Sensor Networks", *In the Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*, pp. 139-152, Boulder, CO, U.S.A, October 2006.

[19] Jayasumana A.P., Han Q., and Illangasekare T., "Virtual Sensor Networks - A Resource Efficient Approach for Concurrent Applications," *In the Proceedings of the 4th International Conference on Information Technology: New Generations (ITNG 2007)*, pp. 111-115, Las Vegas, NV, U.S.A., April 2007.

[20] Manjunatha P., Verma A.K., Srividya A., "Multi-Sensor Data Fusion in Cluster Based Wireless Sensor Networks Using Fuzzy Logic Method", *In the Proceedings of the 3rd IEEE International Conference on Industrial and Information Systems (ICIIS 2008)*, pp. 1-6, Kharagpur, India, December 2008.

[21] Estrin D., Govindan R., Heidemann J., Kumar S., "Next Century Challenges: Scalable Coordination in Sensor Networks," *In the Proceedings the ACM/IEEE International Conference on Mobile Computing and Networking (ACM MobiCom'99)*, pp. 263-270, Seattle, WA, U.S.A., August 1999.

[22] Akyildiz Ian F., Su W., Sankarasubramaniam Y., and Cayirci E., "A Survey on Sensor Networks", *IEEE Communications Magazine*, Vol. 40, No. 8, pp. 102-114, August 2002.

[23] Bonnet P., Gehrke J., Seshadri P., "Querying the Physical World", *IEEE Personal Communications*, Vol. 7, No. 5, pp. 10-15, October 2000.

[24] Alert Users Group, *available at http://www.alertsystems.org*, [Accessed: July 2012].

[25] Ogawa M., Tamura T., Togawa T., "Fully Automated Biosignal Acquisition in Daily Routine Through 1 Month", *In the Proceedings of International Conference on IEEE-EMBS*, pp. 1947-1950, Hong Kong, October 1998.

[26] Johnson P., Andrews D.C., "Remote Continuous Physiological Monitoring in the Home", *Journal of Telemedicine and Telecare*, Vol. 2, pp. 107-113, December 1997.

[27] Celler B.G., Hesketh T., Earnshaw W., Ilsar E., "An Instrumentation System for the Remote Monitoring of Changes in Functional Health Status of the Elderly", *In the Proceedings of the 16th IEEE Annual International Conference of Engineering in Medicine and Biology Society, Engineering Advances: New Opportunities for Biomedical Engineers*, Vol. 2, pp. 908-909, Baltimore, MD, U.S.A, November 1994.

[28] Coyle G., Boydell L., Brown L., "Home Telecare for the Elderly", *Journal of Telemedicine and Telecare*, Vol. 1, No. 3, pp. 183-184, 1995.

[29] Nam Y.H., Zeehun H., Young C.H., Kwang P.S., "Development of Remote Diagnosis System Integrating Digital Telemetry for Medicine," *In the Proceedings of International Conference on IEEE-EMBS*, Vol. 3, pp. 1170-1173, Hong Kong, October 1998.

[30] Ko J., Lim J., Chen Y., Musaloiu-E R., Terzis A., Masson G., Gao T., Destler W., Selavo L., and Dutton R., "MEDiSN: Medical Emergency Detection in Sensor Networks" *In Special Issue on Wireless Health Systems in ACM Transactions on Embedded Computing Systems (TECS)*, Article 11, 29 pages, August 2010.

[31] Bauer P., Sichitiu M., Istepanian R., Premaratne K., "The Mobile Patient: Wireless Distributed Sensor Networks for Patient Monitoring and Care", *In the Proceedings of IEEE EMBS International Conference on Information Technology Applications in Biomedicine*, pp. 17-21, Arlington, VA, U.S.A., November 2000.

[32] Noury N., Herve T., Rialle V., Virone G., Mercier E., Morey G., Moro A., Porcheron T., "Monitoring Behaviour in Home Using a Smart Fall Sensor", *In the Proceedings of IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology*, pp. 607-610, La Tronche, France, October 2000.

[33] Sibbald B., "Use Computerized Systems to Cut Adverse Drug Events", *Journal of Canadian Medical Association,* Vol. 164, No. 13, pp. 1878-1878-a, June 2001.

[34] Petriu E.M., Georganas N.D., Petriu D.C, Makrakis D., and Groza V.Z., "Sensor-Based Information Appliances", *IEEE Instrumentation and Measurement Magazine*, Vol. 3, No. 4, pp. 31-35, December 2000.

[35] Herring C., Kaplan S., "Component-Based Software Systems for Smart Environments", *IEEE Personal Communications*, Vol. 7, No.5, pp. 60-61, October 2000.

[36] Rabaey J.M., Ammer M.J., da Silva Jr. J.L., Patel D., Roundy S., "PicoRadio Supports Adhoc Ultra-Low Power Wireless Networking", *IEEE Journal of Computers*, Vol. 33, No. 7, pp. 42-48, July 2000.

[37] Pottie G.J., Kaiser W.J., "Wireless Integrated Network Sensors", *IEEE Journal of Computers,* Vol. 43, No.5, pp. 551-558, May 2000.

[38] Shih E., Cho S., Ickes N., Min R., Sinha A., Wang A., Chandrakasan A., "Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks", *In the Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pp. 272-286, Rome, Italy, July 2001.

[39] Sustainable Bridges, "Assessment for Future Traffic Demands and Longer Lives", *available at http://www.sustainablebridges.net* [Accessed: July 16, 2012].

[40] Bock Przemek J., Majumdar S., and Bock Wojtek J., "Internet-Based Distributed Data Acquisition System for Fiber-Optic Sensors", *IEEE Transactions on Instrumentation and Measurement*, Vol. 56, No. 1, pp. 32-38, February 2007.

[41] Ramanathan N., Balzano L., Estrin D., Hansen M., Harmon T.C., Jay J.A., Kaiser W.J., and Sukhatme G., "Designing Wireless Sensor Networks as a Shared Resource for Sustainable Development," *In the Proceedings of the 1st International Conference on Information and Communication Technologies and Development (ICTD'06)*, pp. 256-265, Berkeley, CA, U.S.A., May 2006.

[42] Wireless Sensor Network Solutions, "Monitoring Data Centers", *available at http://www.meters.com.hk/Sensicast_home.html* [Accessed: January 7, 2013].

[43] Manjunath D., Gopaliah S.V., "A Multi-purpose Wireless Sensor Network for Residential Layouts," *In the Proceedings of the 9th International Workshop on Information Integration and Web-based Applications and Services (iiWAS '07)*, pp. 49-58, Jakarta, Indonesia, December 2007.

[44] Shen C., Srisathapornphat C., Jaikaeo C., "Sensor Information Networking Architecture and Applications", *IEEE Personal Communications*, Vol. 8, No. 4, pp. 52-59, August 2001.

[45] Perkins C., "Ad Hoc Networks", *Addison-Wesley*, Reading, MA, U.S.A., 2000.

[46] Li L., Halpern J. Y., "Minimum-Energy Mobile Wireless Networks Revisited," *In the Proceedings of IEEE International Conference on Communications (ICC'01)*, pp. 278-283, Helsinki, Finland, June 2001.

[47] Hedetniemi S., Liestman A., "A Survey of Gossiping and Broadcasting in Communication Networks", *International Journal of Networks*, Vol. 18, No. 4, pp. 319–349, 1988.

[48] Heinzelman W.R, Kulik J., Balakrishnan H., "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks", *In the Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, pp. 174–185, Seattle, WA, U.S.A., August 1999.

[49] Sohrabi K., Gao J., Ailawadhi V., Pottie G.J., "Protocols for Self-Organization of a Wireless Sensor Network", *IEEE Personal Communications*, Vol. 7, No. 5, pp. 16-27 October 2000.

[50] Heinzelman W.R., Chandrakasan A., Balakrishnan H., "Energy-Efficient Communication Protocol for Wireless Micro Sensor Networks", *In the Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS-33)*, pp. 1-10, Maui, Hawaii, U.S.A., January 2000.

[51] Intanagonwiwat C., Govindan R., Estrin D., "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *In the Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (Mobi-Com2000)*, pp. 56-67, Boston, MA, U.S.A., August 2000.

[52] Karp, B. and Kung, H.T., "Greedy Perimeter Stateless Routing for Wireless Networks", *In the Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pp. 243-254, Boston, MA, U.S.A., August 2000.

[53] Suriyachai P., Roedig U., Scott A., "A Survey of MAC Protocols for Mission-Critical Applications in Wireless Sensor Networks", *IEEE Communications Surveys Tutorials*, Vol. 14, No. 2, pp. 240-264, 2012.

[54] LAN MAN Standards Committee of the IEEE Computer Society, "IEEE Standard 802.11-1997 Edition", *"IEEE Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification"*, pp. i-445, New York, NY, U.S.A., 1997.

[55] Ye W., Heidemann J, Estrin D., "An Energy-Efficient MAC Protocol for Wireless Sensor Networks", *In the Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, Vol. 3, pp. 1567-1576, New York, NY, U.S.A., June 2002.

[56] Ye W., Heidemann J., and Estrin D., "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks", *IEEE Journal of Transactions on Networking*, Vol. 12, No. 3, pp. 493-506, June 2004.

[57] Dam T., and Langendoen K., "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," *In the Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 171-180, Los Angeles, CA, U.S.A, November 2003.

[58] Lin P., Qiao C., and Wang X., "Medium Access Control with a Dynamic Duty Cycle for Sensor Networks", *In the Proceedings of the IEEE Conference on Wireless Communications and Networking*, Vol. 3, pp. 1534-1539, Atlanta, GA, U.S.A., March 2004.

[59] Adams L., "Capitalizing on 802.11 for Sensor Networks", *White Paper, GainSpan Corporation*, 440 North Wolfe Road, Sunnyvale, CA, U.S.A., February 2008.

[60] GainSpan Inc., "Wi-Fi Sensor Networking Module", 2009, *available at http://www.gainspan.com/news/news_RFM_033109* [Accessed: July 19, 2012].

[61] Lim Hock B., Teo Yong M., Mukherjee P., Lam V., Wong Weng F., See S., "Sensor Grid: Integration of Wireless Sensor Networks and the Grid", *In the Proceedings of the 30th IEEE Conference on Local Computer Networks (LCN '05)*, pp. 91-98, Sydney, Australia, November 2005.

[62] Intanagonwiwat C., Estrin D., Govindan R., and Heidemann J., "Impact of Network Density on Data Aggregation in Wireless Sensor Networks", *In the Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pp. 414-415, Vienna, Austria, July 2002.

[63] Schurgers C., Tsiatsis V., Ganeriwal S., and Srivastava M., "Topology Management for Sensor Networks: Exploiting Latency and Density", *In the Proceedings of the 3rd ACM International Symposium on Mobile Adhoc Networking & Computing (MobiHOC'02)*, pp. 135-145, Lausanne, Switzerland, June 2002.

[64] Yang T. , Zhang S., "Dormancy Scheduling Algorithm Based on Node's Self-Adaptive Density in WSN", *In the Proceedings of the 5th International Joint Conference on INC, IMS and IDC (NCM '09)*, pp. 494-500, Washington, DC, U.S.A., August 2009.

[65] Abawajy J. H., Nahavandi S., and Al-Neyadi F., "Sensor Node Activity Scheduling Approach," *In the Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pp. 72-77, Seoul, Korea, April 2007.

[66] Islam K., Akl S.G., Meijer H., "Maximizing the Lifetime of Wireless Sensor Networks through Domatic Partition", *In the Proceedings of the 34th IEEE Conference on Local Computer Networks (LCN 2009)*, pp. 436-442, Zurich, Switzerland, October 2009.

[67] Roy N., Rajamani V., Julien C., "Supporting Multi-Fidelity-Aware Concurrent Applications in Dynamic Sensor Networks", *In the Proceedings of the 8th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2010)*, pp. 43-49, Mannheim, Germany, March 2010.

[68] Lim H.B., and Lee D., "An Integrated and Flexible Scheduler for Sensor Grids", *In the Proceedings of the 4th International Conference on Ubiquitous Intelligence and Computing (UIC '07)*, pp. 567-578, Hong Kong, China, July 2007.

[69] Na C., Yang Y., and Mishra A., "An Optimal GTS Scheduling Algorithm for Time-Sensitive Transactions in IEEE 802.15.4 Networks", *International Journal of Computer and Telecommunications Networking*, Vol. 52, No. 13, pp. 2543-2557, September 2008.

[70] Reddy T.B., Karthigeyan I., Manoj B.S., and Murthy C.S.R., "Quality of Service Provisioning in Adhoc Wireless Networks: a Survey of Issues and Solutions", *Journal of Adhoc Networks*, Vol. 4, No. 1, pp. 83-124, January 2006.

[71] Abramson D., Giddy J., and Kotler L., "High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?", *In the Proceedings of the 14th International Parallel and Distributed Processing Symposium (IPDPS 2000)*, pp. 520-528, Cancun, Mexico, May 2000.

[72] Lee C., Lehoczky J., Siewiorek D., Rajkumar R., and Hansen J., "A Scalable Solution to the Multi-Resource QoS Problem", *In the Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS 1999)*, pp. 315-326, Phoenix, AZ, U.S.A., December 1999.

[73] Giannecchini S, Caccamo M., and Shih C., "Collaborative Resource Allocation in Wireless Sensor Networks", *In the Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS '04)*, pp. 35-44, Washington, DC, U.S.A, June 2004.

[74] Bian F., Kempe D., and Govindan R., "Utility Based Sensor Selection", *In the Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 11-18, Nashville, TN, U.S.A., April 2006.

[75] Tan J., Tong G., "Dynamic Resource Allocation for Target Tracking in Robotic Sensor Networks", *In the Proceedings of IEEE International Conference on Systems, Man and Cybernetics (SMS 2007)*, pp. 2354-2359, Montreal, QC, Canada, October 2007.

[76] Ren-Shiou L., Sinha P., Koksal C.E., "Joint Energy Management and Resource Allocation in Rechargeable Sensor Networks", *In the Proceedings of 29th IEEE Conference on Communications (INFOCOM 2010)*, pp. 1-9, San Diego, CA, U.S.A, March 2010.

[77] Hariharan S., Bisdikian C., Kaplan L.M., Pham T., "QoI-Based Resource Allocation for Multi-Target Tracking in Energy Constrained Sensor Networks", *In the Proceedings of the 14th International Conference on Information Fusion (FUSION)*, pp. 1-8, Chicago, IL, U.S.A., July 2011.

[78] Bhattacharya S., Saifullah A., Lu C., and Roman G.C., "Multi-Application Deployment in Shared Sensor Networks Based on Quality of Monitoring", *In the Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '10)*, pp. 259-268, Stockholm, Sweden, April 2010.

[79] Curescu C., and Nadjm-Tehrani S., "Price/Utility-Based Optimized Resource Allocation in Wireless Adhoc Networks", *In the Proceedings of the 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2005)*, pp. 85-95, Santa Clara, CA, U.S.A., September 2005.

[80] Ma Q., Parkes D.C., and Welsh M., "A Utility-Based Approach to Bandwidth Allocation and Link Scheduling in Wireless Networks," *In the Proceedings of the 1st International Workshop on Agent Technology for Sensor Networks (ATSN-07)*, Honolulu, Hawaii, May 2007.

[81] Xu Y., Saifullah A., Chen Y., Lu C., and Bhattacharya S., "Near Optimal Multi-Application Allocation in Shared Sensor Networks", *In the Proceedings of the 11th ACM International Symposium on Mobile Adhoc Networking and Computing (MobiHoc '10)*, pp. 181-190, Chicago, IL, U.S.A., September 2010.

[82] Werner-Allen G., Swieskowski P., and Welsh M., "Motelab: A Wireless Sensor Network Testbed", *In the Proceedings of the 4th International Conference on Information Processing in Sensor Networks (ISPN '05), Special Track on Platform Tools and Design*

Methods for Network Embedded Sensors (SPOTS), pp. 73-78, Los Angeles, CA, U.S.A., April 2005.

[83] Girod L., Elson J., Cerpa A., Stathopoulos T., Ramanathan N., and Estrin D., "Em*: A Software Environment for Developing and Deploying Wireless Sensor Networks", *Technical Report 0034*, Center for Embedded Networked Sensing, Computer Science Department, University of California, Los Angeles, CA, U.S.A., December 2003.

[84] Sridharan M., Bapat S., Ramnath R., and Arora. A., "Implementing an Autonomic Architecture for Fault-Tolerance in a Wireless Sensor Network Testbed for At-Scale Experimentation", *In the Proceedings of the 2008 ACM Symposium on Applied Computing (SAC '08)*, pp. 1670-1676, Fortaleza, Ceara, Brazil, March 2008.

[85] Bramley R., Chiu K., Huffman J.C., Huffman K., and McMullen D.F., "Instruments and Sensors as Network Services: Making Instruments First Class Members of the Grid", *Technical Report 588*, School of Informatics and Computing, Indiana University, Bloomington, IN, U.S.A., December 2003.

[86] Foster I., and Kesselman C., "Globus: A Metacomputing Infrastructure Toolkit", *International Journal of Supercomputer Applications*, Vol. 11, No. 2, pp. 115-128, 1997.

[87] Open Grid Service Infrastructure WG, Global Grid Forum, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", 2002, *available at http://www.globus.org/alliance/publications/papers/ogsa.pdf*, [Accessed: July 2012].

[88] Sobeih A., Chen W., Hou Jennifer C., Kung L., Li N., Lim H., Tyan H., and Zhang H., "J-Sim: A Simulation Environment for Wireless Sensor Networks", *In the Proceedings of the 38th Annual Simulation Symposium (ANSS'05)*, pp. 175-187, San Diego, CA, U.S.A., April 2005.

[89] Park S., Savvides A., and Srivastava Mani B., "SensorSim: A Simulation Framework for Sensor Networks", *In the Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM '00)*, pp. 104-111, Boston, MA, U.S.A., August 2000.

[90] Park S., Savvides A., and Srivastava Mani B., "Simulating Networks of Wireless Sensors", *In the Proceedings of the 2001 Winter Simulation Conference (WSC 2001)*, Vol. 2, pp. 1330-1338, Arlington, VA, U.S.A., December 2001.

[91] Shen C., Srisathapornphat C., Jaikaeo C., "Sensor Information Networking Architecture and Applications", *IEEE Personal Communications*, Vol. 8, No. 4, pp. 52–59, August 2001.

[92] Kumar D., and Patel R.B., "Multi-Hop Data Communication Algorithm for Clustered Wireless Sensor Networks", *International Journal of Distributed Sensor Networks*, Vol. 2011 Article ID 984795, 10 pages, 2011.

[93] Wang Y., Hu C., Tseng Y., "Efficient Placement and Dispatch of Sensors in a Wireless Sensor Network", *IEEE Journal of Transactions on Mobile Computing*, Vol.7, No. 2, pp. 262-274, February 2008.

[94] Bai X., Yun Z., Xuan D., Lai T.H., Jia W., "Optimal Patterns for Four-Connectivity and Full Coverage in Wireless Sensor Networks", *IEEE Journal of Transactions on Mobile Computing*, Vol. 9, No. 3, pp. 435-448, March 2010.

[95] Poe W.Y., Schmitt J.B., "Node Deployment in Large Wireless Sensor Networks: Coverage, Energy Consumption, and Worst-Case Delay", *In the Proceedings of the Asian Internet Engineering Conference*, pp. 77-84, Bangkok, Thailand, November 2009.

[96] Zheng J., Jamalipour A., "Wireless Sensor Networks: A Networking Perspective," *John Wiley & Sons*, Hoboken, NJ, U.S.A., 2009.

[97] Chatterjea S., De Luigi S., Havinga P., "An Adaptive Directed Query Dissemination Scheme for Wireless Sensor Networks", *In the Proceedings of International Conference on Parallel Processing Workshops (ICPPW 2006)*, pp. 181-188, Columbus, OH, U.S.A., August 2006.

[98] Benenson Z., Bestehorn M., Buchmann E., Freiling F. C.., and Jawurek M., "Query Dissemination with Predictable Reachability and Energy Usage in Sensor Networks", *In the Proceedings of the 7th International Conference on Ad-hoc, Mobile and Wireless Networks (ADHOC-NOW '08)*, pp. 279-292, Sophia-Antipolis, France, September 2008.

[99] Liu X., Huang Q., Zhang Y., "Balancing Push and Pull for Efficient Information Discovery in Large-Scale Sensor Networks", *IEEE Journal of Transactions on Mobile Computing*, Vol. 6, No.3, pp. 241-251, March 2007.

[100] Kapoor N., Majumdar S., and Nandy B., "Scheduling on Wireless Sensor Network Hosting Multiple Applications", *In the Proceedings of IEEE International Conference on Communications (ICC 2011)*, doi: 10.1109/icc.2011.5963195, pp. 1-6, Kyoto, Japan, June 2011.

[101] Zimmermann H., "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection", *IEEE Journal of Transactions on Communications*, Vol. 28, No. 4, pp. 425-432, April 1980.

[102] J. De Couto D. S., Aguayo D., Bicket J., and Morris R., "A High-Throughput Path Metric for Multi-Hop Wireless Networks", *Journal of Wireless Networks*, Vol. 11, No. 4, pp. 419-434, July 2005.

[103] Li J., Blake C., J. De Couto D.S., Lee H. I., and Morris R., "Capacity of Adhoc Wireless Networks", *In the Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, pp. 61-69, Rome, Italy, July 2001.

[104] Zhai H., and Fang Y., "A Distributed Adaptive Packet Concatenation Scheme for Sensor and Ad Hoc Networks", *In the Proceedings of the IEEE Military Communications Conference (Milcom'05)*, Vol. 3, pp. 1443-1449, Atlantic City, NJ, U.S.A., October 2005.

[105] Yun L., Roychoudhury A., Mitra T., "Timing Analysis of Body Area Network Applications", *In the Proceedings of the Conference on Worst-Case Execution Time Analysis (WCET 2007)*, Vol. 6, pp. 1192-1197, Pisa, Italy, July 2007.

[106] Kapoor N., Majumdar S., and Nandy B., "System and Application Knowledge Based Scheduling of Multiple Applications in a WSN", *In the Proceedings of IEEE International Conference on Communications (ICC 2012)*, pp. 355-360, Ottawa, ON, Canada, June 2012.

[107] Poe W.Y., Schmitt J.B., "Node Deployment in Large Wireless Sensor Networks: Coverage, Energy Consumption, and Worst-Case Delay", *In the Proceedings of the Asian Internet Engineering Conference*, pp. 77-84, Bangkok, Thailand, November 2009.

[108] Shnayder V., Hempstead M., Chen B, Allen G. W., and Welsh M., "Simulating the Power Consumption of Large-Scale Sensor Network Applications", *In the Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 188-200, Baltimore, MD, U.S.A, November 2004.

[109] Kapoor N., Majumdar S., and Nandy B., "Sensor Allocation to Multiple Applications in Shared Wireless Sensor Networks", Extended Abstract for Poster, *In the Proceedings of 31st IEEE International Performance Computing and Communications Conference (IPCCC 2012)*, pp. 197-198, Austin, TX, U.S.A., December 2012.

APPENDIX A: Pseudo Code of Scheduling Algorithms

A.1 Least Number Distance Product First (LNDPF)

```
    Initialize
    Received Requests = NULL; /* List of requests received by the proxy */
    integer i;
    boolean flag_added = false; /* Indicates whether the request has been added*/
    Enqueue (Upon Arrival of Application Request at the Proxy)
if (!ReceivedRequests.isEmpty())/* if the list containing incoming requests is not empty*/
        for ( i = 0 ; i< ReceivedRequests.size(); i++) /* Scan the list */
                if (NDP of application for incoming request < NDP of application for
request in list)
                        /*Insert the request at the current position in the list*/
                                ReceivedRequests.add (i, ReceivedUserRequest);
                                flag_added = true;
                                break; /* exit the for loop*/
                end if
                if (NDP of incoming request == NDP of request in list)
/*Scan the list to the point where the NDP of the application for the request in the list is
greater than the NDP of the application for the incoming request*/
                                int j;
                        for ( j = i+1 ; j< ReceivedRequests.size(); j++)
                                /*Get the NDP of the application for the request in the
                                list*/
                                if (NDP of request in list> NDP of incoming request)
                                        /*Place the current request at the current position
                                        in the list*/
                                        ReceivedRequests.add (j, ReceivedUserRequest);
                                        flag_added = true;
                                        break;
                                end if
                        end for
                end if
        end for
                if (!flag_added) /*if the request is not added to the list*/
                ReceivedRequests.add(ReceivedRequests.size(), ReceivedUserRequest);
                end if
end if
if (ReceivedRequests.isEmpty())
        /*Add the Received Request to Received Requests*/
        ReceivedRequests.add(0, ReceivedUserRequest);
end if
```

Dequeue

*/*Get the requests in order of non-decreasing NDP from the ordered list*/*

while (!ReceivedRequests.isEmpty()) {

*/*While there are still some requests to be processed, send them to the WSN through the wireless protocol stack */*

ReceivedRequests.remove (0);

Send request to WSN

Remove request from the list

end while

A.2 Least Farthest Number Distance Product First (LFNDPF)

```
    Initialize
    Received Requests = NULL; /* List of requests received by the proxy */
    integer i;
    boolean flag_added = false; /* Indicates whether the request has been added */
    Enqueue (Upon Arrival of Application Request at the Proxy)
if (!ReceivedRequests.isEmpty())/* if the list containing incoming requests is not empty*/
        for ( i = 0 ; i< ReceivedRequests.size(); i++) /* Scan the list */
                if (FNDP of application for incoming request < FNDP of application for
request in list)
                        /*Insert the request at the current position in the list*/
                                ReceivedRequests.add (i, ReceivedUserRequest);
                                flag_added = true;
                                break; /* exit the for loop*/
                end if
                if (FNDP of incoming request == FNDP of request in list)
/* Scan the list to the point where the FNDP of the application for the request in the list
is greater than the FNDP of the application for the incoming request*/
                        int j;
                        for ( j = i+1 ; j< ReceivedRequests.size(); j++)
                                /*Get the FNDP of the application for the request in the
                                list*/
                                if (FNDP of request in list> FNDP of incoming request)
                                        /*Place the current request at the current position
                                        in the list*/
                                        ReceivedRequests.add (j,ReceivedUserRequest);
                                        flag_added = true;
                                        break;
                                end if
                        end for
                end if
        end for
                if (!flag_added) /*if the request is not added to the list */
                /* Add the request to the end of the list*/
                ReceivedRequests.add(ReceivedRequests.size(), ReceivedUserRequest);
                end if
end if /* end of if (!ReceivedRequests.isEmpty())*/
if (ReceivedRequests.isEmpty())
        /*Add the Received Request to Received Requests*/
        ReceivedRequests.add(0, ReceivedUserRequest);
end if
```

Dequeue

*/*Get the requests in order of non-decreasing FNDP from the ordered list*/*

while (!ReceivedRequests.isEmpty()) {

*/*While there are still some requests to be processed, send them to the WSN through the wireless protocol stack */*

ReceivedRequests.remove(0);

Send request to WSN

Remove request from the list

end while

A.3 Least Weighted Farthest Number Distance Product First (LWFNDPF)

```
    Initialize
    Received Requests = NULL; /* List of requests received by the proxy */
    integer i;
    boolean flag_added = false;  /* Indicates whether the request has been added */
    Enqueue (Upon Arrival of Application Request at the Proxy)
if (!ReceivedRequests.isEmpty())/* if the list containing incoming requests is not empty*/
        for ( i = 0 ; i< ReceivedRequests.size(); i++) /* Scan the list */
                if (WFNDP of application for incoming request < WFNDP of application
for request in list)
                        /*Insert the request at the current position in the list*/
                                ReceivedRequests.add (i, ReceivedUserRequest);
                                flag_added = true;
                                break;  /* exit the for loop*/
                end if
                if (WFNDP of incoming request == WFNDP of request in list)
/* Scan the list to the point where the WFNDP of the application for the  request in the
list is greater than the WFNDP of the application for the  incoming request */
                                int j;
                        for ( j = i+1 ; j< ReceivedRequests.size(); j++)
                                /*Get the WFNDP of the  application for the request in the
                                list*/
                                if (WFNDP of request in list> WFNDP of incoming
                                request)
                                        /*Place the current request at the current position
                                        in the list*/
                                        ReceivedRequests.add (j,ReceivedUserRequest);
                                        flag_added = true;
                                        break;
                                end if
                        end for
                end if
        end for
                if (!flag_added) /*if the request is not added to the list*/
                /* Add the request to the end of the list*/
                ReceivedRequests.add(ReceivedRequests.size(), ReceivedUserRequest);
end if
if (ReceivedRequests.isEmpty())
        /*Add the Received Request to Received Requests*/
        ReceivedRequests.add(0, ReceivedUserRequest);
end if
```

Dequeue

*/*Get the requests in order of non-decreasing WFNDP from the ordered list*/*

while (!ReceivedRequests.isEmpty()) {

*/*While there are still some requests to be processed, send them to the WSN through the wireless protocol stack */*

ReceivedRequests.remove (0);

Send request to WSN

Remove request from the list

end while

APPENDIX B: Pseudo Code of Allocation Algorithms

B.1 Pseudo Code of Static Allocation Algorithm

B.1.1 Random Allocation (RA)

Allocate Sensor Nodes to an Application

```
    /* If the application requires sensors at 1 hop distance, select the required
number of sensors from the list at random */
     if (No_of_Sensors_1hop>0)
          for (i=0;i< No_of_Sensors_1hop;i++)
          Select a sensor node from the list at random
          end for
    /* If the application requires sensors at 2 hop distance, select the required
number of sensors from the list at random */
     if (No_of_Sensors_2hop>0)
          for (i=0;i< No_of_Sensors_2hop;i++)
          Select a sensor node from the list at random
          end for
    /* If the application requires sensors at 3 hop distance, select the required
number of sensors from the list at random */
     if (No_of_Sensors_3hop>0)
          for (i=0;i< No_of_Sensors_3hop;i++)
          Select a sensor node from the list at random
          end for
    /* If the application requires sensors at 4 hop distance, select the required
number of sensors from the list at random */
     if (No_of_Sensors_4hop>0)
          for (i=0;i< No_of_Sensors_4hop;i++)
          Select a sensor node from the list at random
          end for
```

Send Request to WSN

```
  /*Get the sensor nodes that are allocated to a particular application*/
for (int j = 1; j < sensorsList.size(); j++)
        {
    SensorNode sensorNode = (SensorNode) sensorsList.get(j);
    SensorDetail detail = sensorNode.getDetail();
       if (Sensor Node is allocated to the Application)
             {
             Send the request to the sensor node
             }
end for
```

B.1.2 CPU Load Balanced Allocation (CLBA)

Allocate Sensor Nodes to an Application

/ For each hop, check If the application requires sensors at the given hop distance, select the required number of sensors from a list sorted on the basis of cpu load. Allocate the sensor nodes and update the cpu load metric. */*

```
if (No_of_Sensors_1hop>0)
Collections.sort(sensorDetails_1hop,new ExecutionTimeComparator());
    for (i=0;i< No_of_Sensors_1hop;i++)
        detail = (SensorDetail) sensorDetails_1hop.get(i)
        int Sensor_id = detail.getSensorId()
        detail.app_ids.add(Integer.toString(Application.getApplicationId()));
        detail.execution_time_sum = detail.execution_time_sum +
        Application.getExecutionTime();
    end for
    if (No_of_Sensors_2hop>0)
Collections.sort(sensorDetails_2hop,new ExecutionTimeComparator());
    for (i=0;i< No_of_Sensors_2hop;i++)
        detail = (SensorDetail) sensorDetails_2hop.get(i)
        int Sensor_id = detail.getSensorId()
        detail.app_ids.add(Integer.toString(Application.getApplicationId()));
        detail.execution_time_sum = detail.execution_time_sum +
        Application.getExecutionTime();
    end for
    if (No_of_Sensors_3hop>0)
Collections.sort(sensorDetails_3hop,new ExecutionTimeComparator());
    for (i=0;i< No_of_Sensors_3hop;i++)
        detail = (SensorDetail) sensorDetails_3hop.get(i)
        int Sensor_id = detail.getSensorId()
        detail.app_ids.add(Integer.toString(Application.getApplicationId()));
        detail.execution_time_sum = detail.execution_time_sum +
        Application.getExecutionTime();
    end for
    if (No_of_Sensors_4hop>0)
Collections.sort(sensorDetails_4hop,new ExecutionTimeComparator());
    for (i=0;i< No_of_Sensors_4hop;i++)
        detail = (SensorDetail) sensorDetails_1hop.get(i)
        int Sensor_id = detail.getSensorId()
        detail.app_ids.add(Integer.toString(Application.getApplicationId()));
        detail.execution_time_sum = detail.execution_time_sum +
        Application.getExecutionTime();
    end for
```

Send Request to WSN

```
  /*Get the sensor nodes that are allocated to a particular application*/
for (int j = 1; j < sensorsList.size(); j++)
        {
    SensorNode sensorNode = (SensorNode) sensorsList.get(j);
    SensorDetail detail = sensorNode.getDetail();
        if (Sensor Node is allocated to the Application)
                {
                Send the request to the sensor node
                }
end for
```

B.1.3 Data Load Balanced Allocation (DLBA)

Allocate Sensor Nodes to an Application

/ /* For each hop, check If the application requires sensors at the given hop distance, select the required number of sensors from a list sorted on the basis of data load on the sensor nodes Allocate the sensor nodes and update the data load metric. */*

```
if (No_of_Sensors_1hop>0)
Collections.sort(sensorDetails_1hop,new DataComparator());
for (i=0;i< No_of_Sensors_1hop;i++)
        detail = (SensorDetail) sensorDetails_1hop.get(i)
        int Sensor_id = detail.getSensorId()
        detail.app_ids.add(Integer.toString(Application.getApplicationId()));
        detail.data_size_sum = detail.data_size_sum + Application.data_size;
end for
if (No_of_Sensors_2hop>0)
Collections.sort(sensorDetails_2hop,new DataComparator());
for (i=0;i< No_of_Sensors_2hop;i++)
        detail = (SensorDetail) sensorDetails_2hop.get(i)
        int Sensor_id = detail.getSensorId()
        detail.app_ids.add(Integer.toString(Application.getApplicationId()));
        detail.data_size_sum = detail.data_size_sum + Application.data_size;
end for
if (No_of_Sensors_3hop>0)
Collections.sort(sensorDetails_3hop,new DataComparator());
for (i=0;i< No_of_Sensors_3hop;i++)
        detail = (SensorDetail) sensorDetails_3hop.get(i)
        int Sensor_id = detail.getSensorId()
        /* Allocate the sensor nodes and update the data load metric */
        detail.app_ids.add(Integer.toString(Application.getApplicationId()));
        detail.data_size_sum = detail.data_size_sum + Application.data_size;
end for
if (No_of_Sensors_4hop>0)
Collections.sort(sensorDetails_4hop,new DataComparator());
for (i=0;i< No_of_Sensors_4hop;i++)
        detail = (SensorDetail) sensorDetails_4hop.get(i)
        int Sensor_id = detail.getSensorId()
        /* Allocate the sensor nodes and update the data load metric */
        detail.app_ids.add(Integer.toString(Application.getApplicationId()));
        detail.data_size_sum = detail.data_size_sum + Application.data_size;
end for
```

Send Request to WSN

```
  /*Get the sensor nodes that are allocated to a particular application*/
for (int j = 1; j < sensorsList.size(); j++)
        {
    SensorNode sensorNode = (SensorNode) sensorsList.get(j);
    SensorDetail detail = sensorNode.getDetail();
        if (Sensor Node is allocated to the Application)
                {
                Send the request to the sensor node
                }
end for
```

B.1 Pseudo Code of Dynamic Allocation Algorithms

B.2.1 Dynamic Random Allocation (DRA)

```
    Send Request to WSN
/* For each hop distance, select the required number of sensors at random. */
        if (No_of_Sensors_1hop>0)
        for(int i = 0 ;i<Sensors_1hop;i++)
                Random rand = new Random()
                detail = (SensorDetail)
                sensorDetails_1hop.get(rand.nextInt(sensorDetails_1hop.size()));
                int Sensor_id = detail.getSensorId();
                Sensors_1hop_AlreadyAllocated.add(detail);
                Send the request to the allocated sensor node
        end for
        if (No_of_Sensors_2hop>0)
        for(int i = 0 ;i<Sensors_2hop;i++)
                Random rand = new Random()
                detail = (SensorDetail)
                sensorDetails_2hop.get(rand.nextInt(sensorDetails_2hop.size()));
                int Sensor_id = detail.getSensorId();
                Sensors_2hop_AlreadyAllocated.add(detail);
                Send the request to the allocated sensor node
        end for
        if (No_of_Sensors_3hop>0)
        for(int i = 0 ;i<Sensors_3hop;i++)
                Random rand = new Random()
                detail = (SensorDetail)
                sensorDetails_3hop.get(rand.nextInt(sensorDetails_3hop.size()));
                int Sensor_id = detail.getSensorId();
                Sensors_3hop_AlreadyAllocated.add(detail);
                Send the request to the allocated sensor node
        end for
        if (No_of_Sensors_4hop>0)
        for(int i = 0 ;i<Sensors_4hop;i++)
                Random rand = new Random()
                detail = (SensorDetail)
                sensorDetails_4hop.get(rand.nextInt(sensorDetails_4hop.size()));
                int Sensor_id = detail.getSensorId();
                Sensors_4hop_AlreadyAllocated.add(detail);
                Send the request to the allocated sensor node
        end for
```

B.2.2 Dynamic CPU Load Balanced Allocation (DCLBA)

Initialize

/ For all sensor nodes, initialize the sum of execution time of executed requests*/*

execution_time_sum = 0

Send Request to WSN

/ For each hop distance, sort the list containing sensor nodes based on the sum of execution time of applications allocated to the sensor node. Select the sensor nodes with lower value of this sum and update this sum */*

```
if (No_of_Sensors_1hop>0)
Collections.sort (sensorDetails_1hop, new ExecutionTimeComparator());
for (i=0;i< No_of_Sensors_1hop;i++)
        detail = (SensorDetail) sensorDetails_1hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
        detail.execution_time_sum = detail.execution_time_sum +
        Request_Application.getExecutionTime()
end for
if (No_of_Sensors_2hop>0)
Collections.sort(sensorDetails_2hop,new ExecutionTimeComparator());
for (i=0;i< No_of_Sensors_2hop;i++)
        detail = (SensorDetail) sensorDetails_2hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
        detail.execution_time_sum = detail.execution_time_sum +
        Request_Application.getExecutionTime()
end for
if (No_of_Sensors_3hop>0)
Collections.sort(sensorDetails_1hop,new ExecutionTimeComparator());
for (i=0;i< No_of_Sensors_3hop;i++)
        detail = (SensorDetail) sensorDetails_3hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
        detail.execution_time_sum = detail.execution_time_sum +
        Request_Application.getExecutionTime()
end for
if (No_of_Sensors_4hop>0)
Collections.sort(sensorDetails_4hop,new ExecutionTimeComparator());
for (i=0;i< No_of_Sensors_4hop;i++)
        detail = (SensorDetail) sensorDetails_1hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
        detail.execution_time_sum = detail.execution_time_sum +
        Request_Application.getExecutionTime()
end for
```

B.2.3 Dynamic Data Load Balanced Allocation (DDLBA)

Initialize

/ For all sensor nodes, initialize the sum of data size of response messages transmitted by a sensor node */*

```
data_size_sum = 0
```

Send Request to WSN

/ For each hop distance, sort the list containing sensor nodes based on the sum of size of response messages corresponding to requests allocated to them . Select the sensor nodes with a lower value of this sum and update the sum of response messages of executed requests that are transmitted by a sensor node */*

```
if (No_of_Sensors_1hop>0)
Collections.sort(sensorDetails_1hop,new DataComparator());
for (i=0;i< No_of_Sensors_1hop;i++)
        detail = (SensorDetail) sensorDetails_1hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
        detail.data_size_sum = detail.data_size_sum +
        Request_Application.data_size
end for
if (No_of_Sensors_2hop>0)
Collections.sort(sensorDetails_2hop,new DataComparator());
for (i=0;i< No_of_Sensors_2hop;i++)
        detail = (SensorDetail) sensorDetails_2hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
        detail.data_size_sum = detail.data_size_sum +
        Request_Application.data_size
end for
if (No_of_Sensors_3hop>0)
Collections.sort(sensorDetails_3hop,new DataComparator());
for (i=0;i< No_of_Sensors_3hop;i++)
        detail = (SensorDetail) sensorDetails_3hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
        detail.data_size_sum = detail.data_size_sum +
        Request_Application.data_size
end for
if (No_of_Sensors_4hop>0)
Collections.sort(sensorDetails_1hop,new DataComparator());
for (i=0;i< No_of_Sensors_4hop;i++)
        detail = (SensorDetail) sensorDetails_1hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
        detail.data_size_sum = detail.data_size_sum +
        Request_Application.data_size
end for
```

B.2.4 Balanced Metric Allocation (BMA)

Initialize

/ For all sensor nodes, initialize the metric corresponding to both the CPU load and the load on the radio component of the sensor nodes */*

metric_sum = 0

Send Request to WSN

/ If the application requires sensors at 1 hop distance, sort the list containing sensor nodes based on the balanced metric and select the nodes with lower value of metric. */*

```
if (No_of_Sensors_1hop>0)
 Collections.sort(sensorDetails_1hop,new MetricComparator());
for (i=0;i< No_of_Sensors_1hop;i++)
        detail = (SensorDetail) sensorDetails_1hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
end for
```

/ If the application requires sensors at 2 hop distance, sort the list containing sensor nodes based on the balanced metric and select the nodes with lower value of metric. */*

```
if (No_of_Sensors_2hop>0)
 Collections.sort(sensorDetails_1hop,new MetricComparator());
for (i=0;i< No_of_Sensors_2hop;i++)
        detail = (SensorDetail) sensorDetails_2hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
end for
```

/ If the application requires sensors at 3 hop distance, sort the list containing sensor nodes based on the balanced metric and select the nodes with lower value of metric.*/*

```
if (No_of_Sensors_3hop>0)
 Collections.sort(sensorDetails_3hop,new MetricComparator());
for (i=0;i< No_of_Sensors_3hop;i++)
        detail = (SensorDetail) sensorDetails_1hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
end for
```

/ If the application requires sensors at 4 hop distance, sort the list containing sensor nodes based on the balanced metric and select the nodes with lower value of metric.*/*

```
if (No_of_Sensors_4hop>0)
 Collections.sort(sensorDetails_4hop,new MetricComparator());
for (i=0;i< No_of_Sensors_4hop;i++)
        detail = (SensorDetail) sensorDetails_4hop.get(i);
        int Sensor_id = detail.getSensorId();
        Send the request to the allocated sensor node
end for
```

B.2.5 Maximum Energy First (MEF)

Initialize

/ For all sensor nodes, initialize the energy*/*

energy = 1000 Joules */* For sensor nodes with equal amount of initial energy or set the values according to a distribution*/*

Send Request to WSN

/ Check if the application requires sensors at 1 hop distance */*

if (No_of_Sensors_1hop>0)

/ Select the nodes with higher energy level as compared to the other sensor nodes */*

for (i=0;i< No_of_Sensors_1hop;i++)

Amongst the unallocated nodes, get the sensor node with the highest level of energy

Send the request to the allocated sensor node

end for

/ Check if the application requires sensors at 2 hop distance */*

if (No_of_Sensors_2hop>0)

/ Select the nodes with higher energy level as compared to the other sensor nodes */*

for (i=0;i< No_of_Sensors_2hop;i++)

Amongst the unallocated nodes, get the sensor node with the highest level of energy

Send the request to the allocated sensor node

end for

/ Check if the application requires sensors at 3 hop distance */*

if (No_of_Sensors_3hop>0)

/ Select the nodes with higher energy level as compared to the other sensor nodes */*

for (i=0;i< No_of_Sensors_3hop;i++)

Amongst the unallocated nodes, get the sensor node with the highest level of energy

Send the request to the allocated sensor node

end for

/ Check if the application requires sensors at 4 hop distance */*

if (No_of_Sensors_4hop>0)

/ Select the nodes with higher energy level as compared to the other sensor nodes */*

for (i=0;i< No_of_Sensors_1hop;i++)

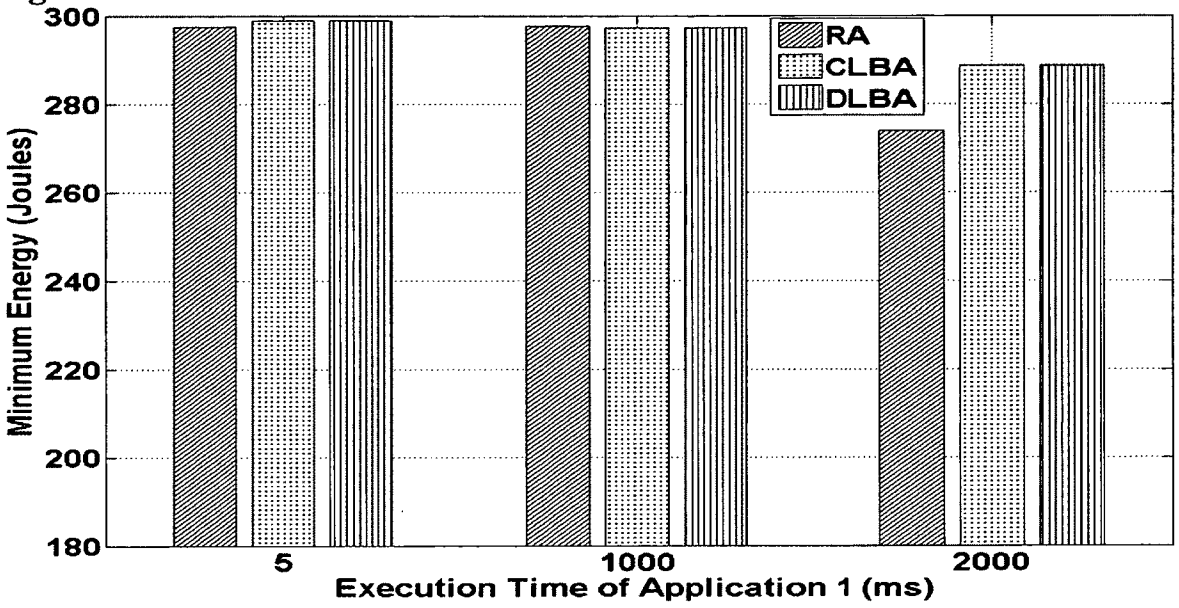
Amongst the unallocated nodes, get the sensor node with the highest level of energy

Send the request to the allocated sensor node

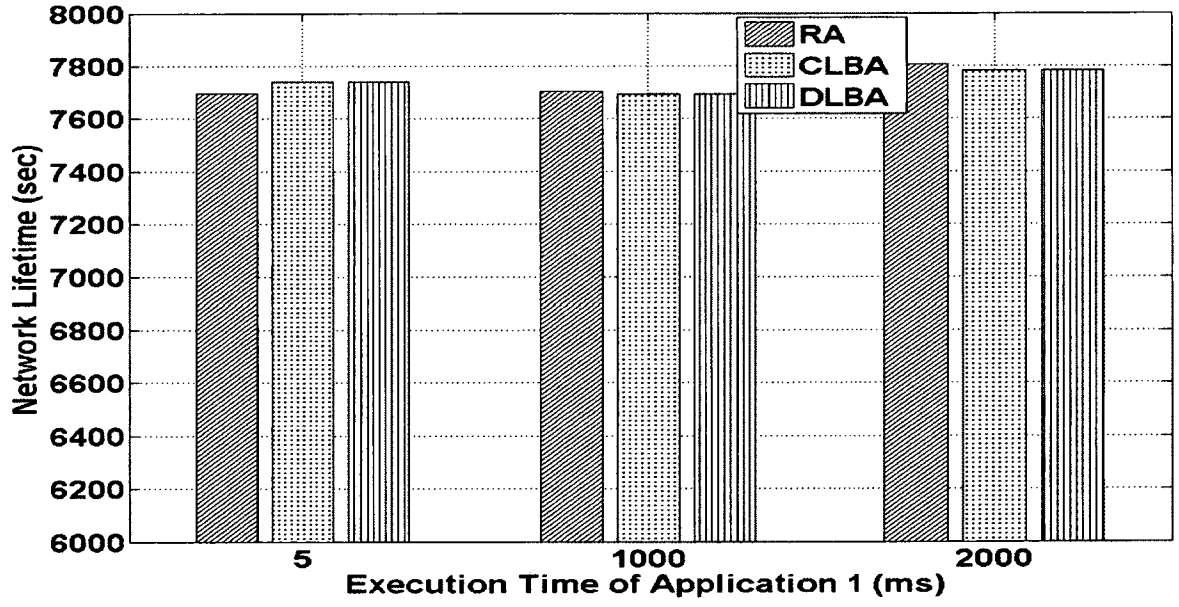
end for

Appendix C: Performance of Allocation Algorithms for Sensor Nodes with Unequal Initial Energy

C.1: Effect of Execution Time on the Performance of Static Allocation Algorithms



(a)

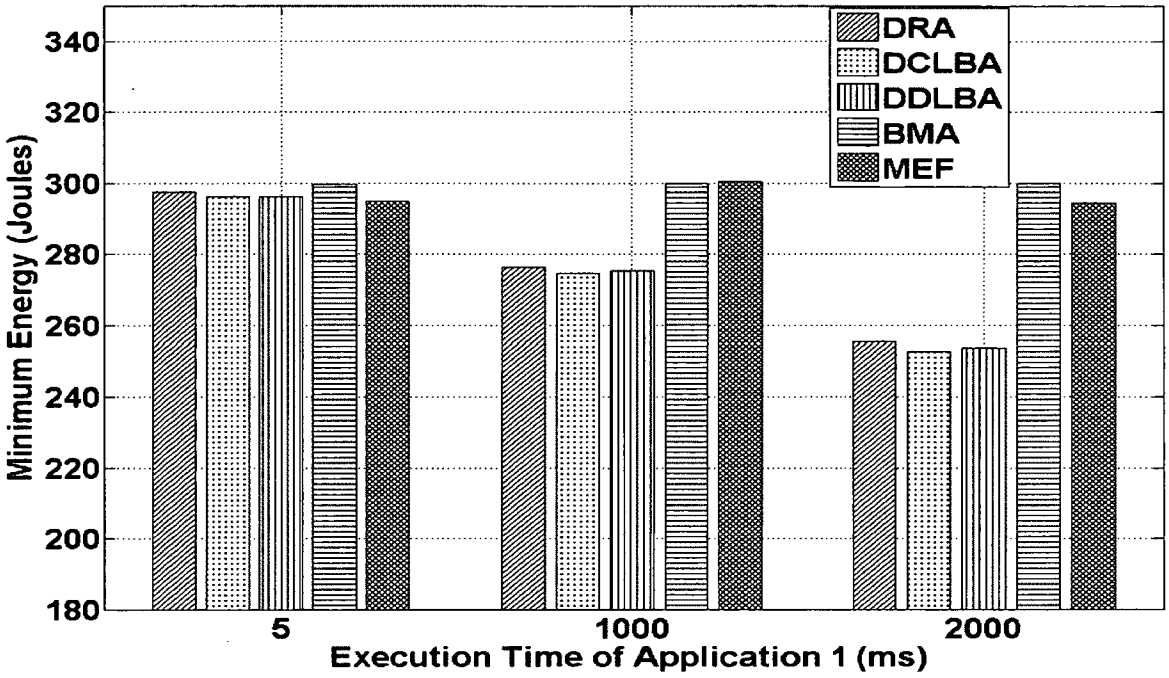


(b)

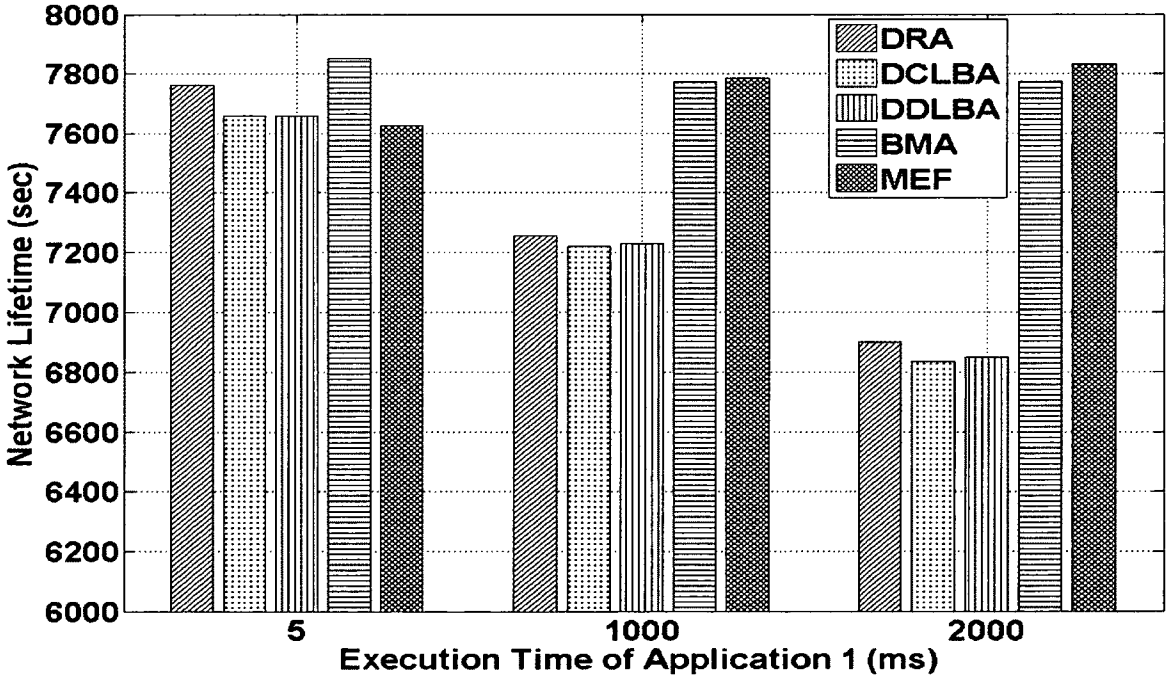
Figure C-1: Effect of Execution Time on the Performance of Static Allocation Algorithms with Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

C.2: Effect of Execution Time on the Performance of Dynamic Allocation Algorithms



(a)

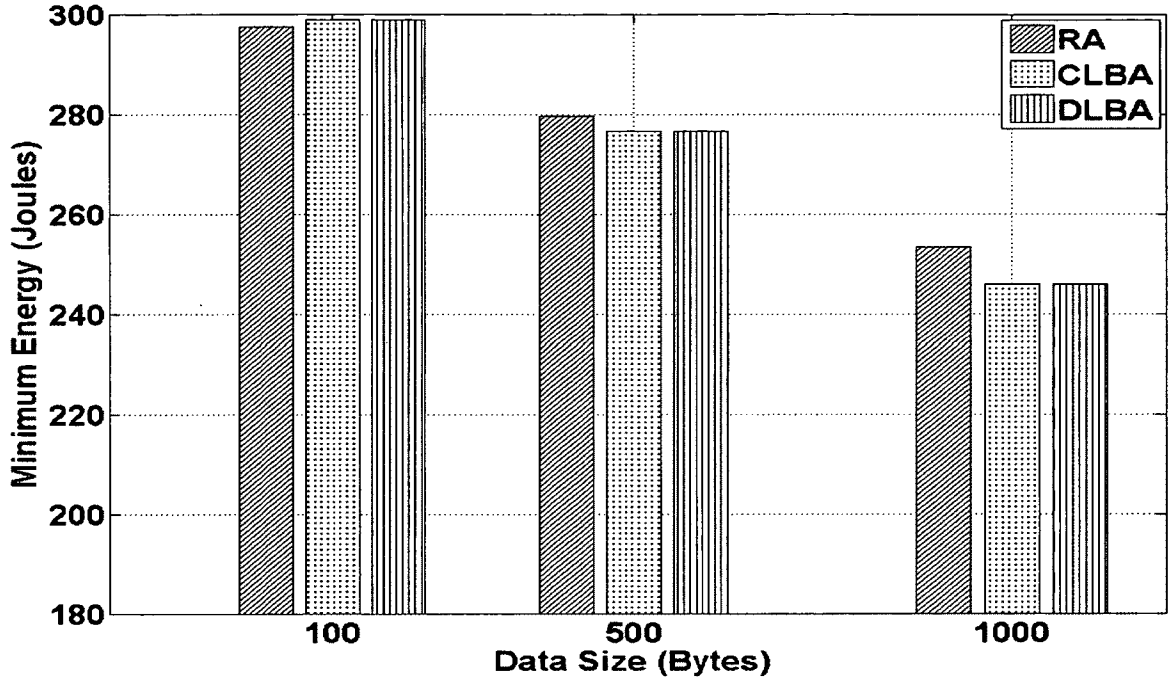


(b)

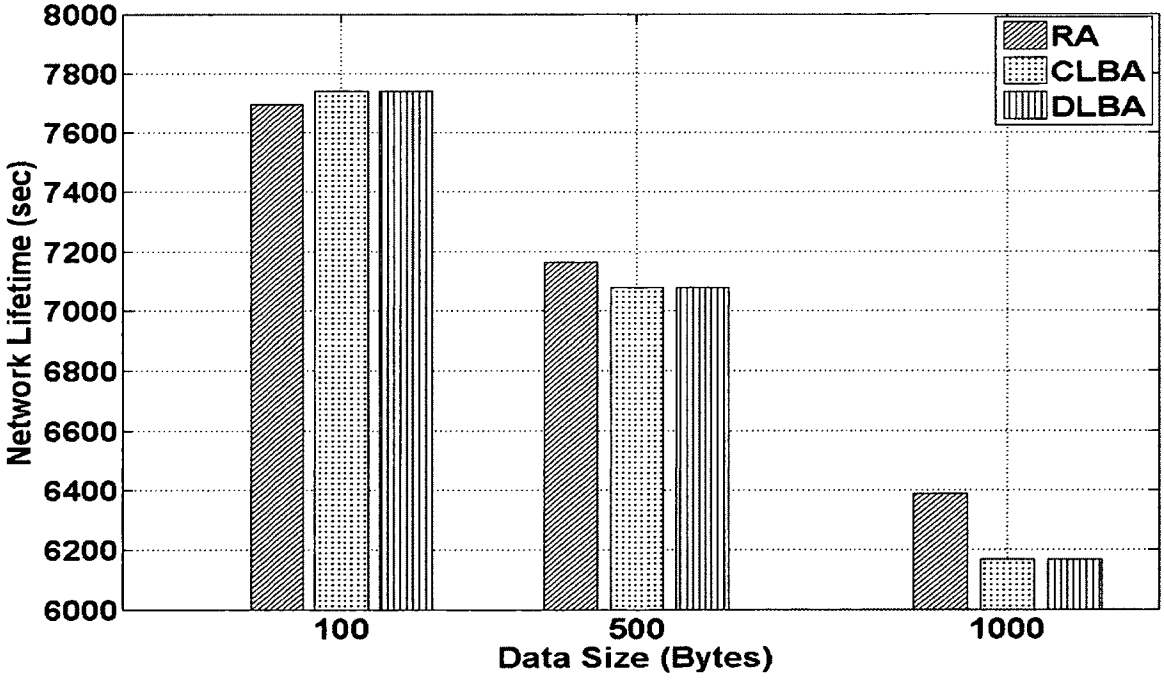
Figure C-2: Effect of Execution Time on the Performance of Dynamic Allocation Algorithms with Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

C.3: Effect of Data Size on the Performance of Static Allocation Algorithms



(a)

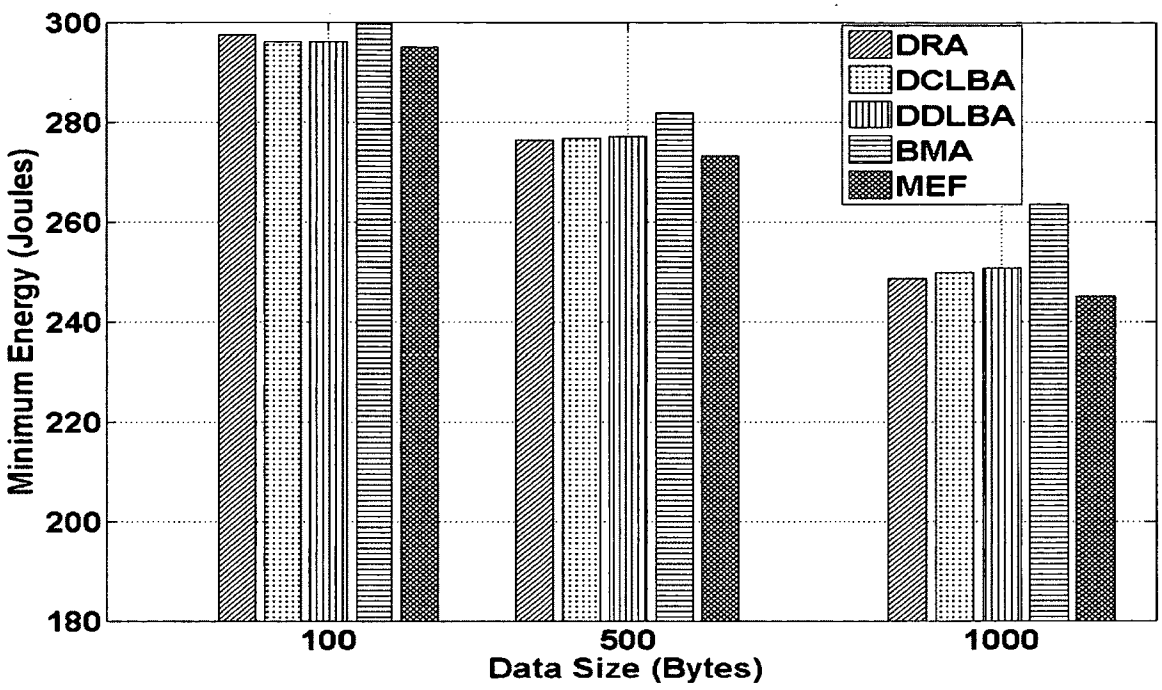


(b)

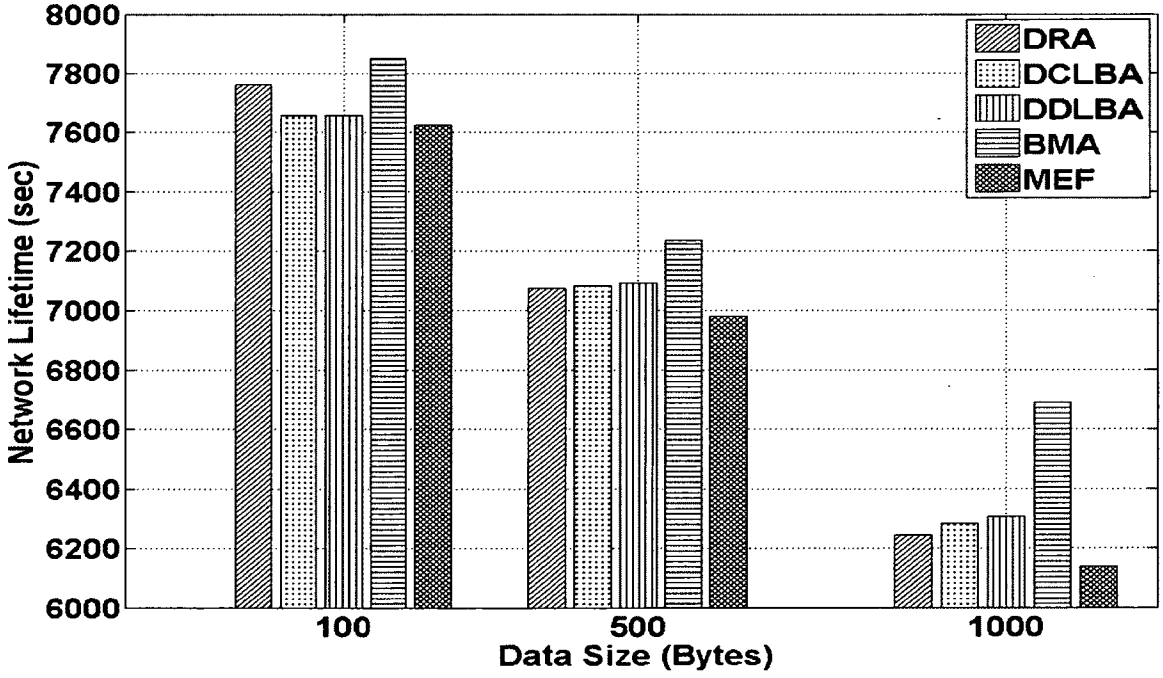
Figure C-3: Effect of Data Size on the Performance of Static Allocation Algorithms with Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

C.4: Effect of Data Size on the Performance of Dynamic Allocation Algorithms



(a)

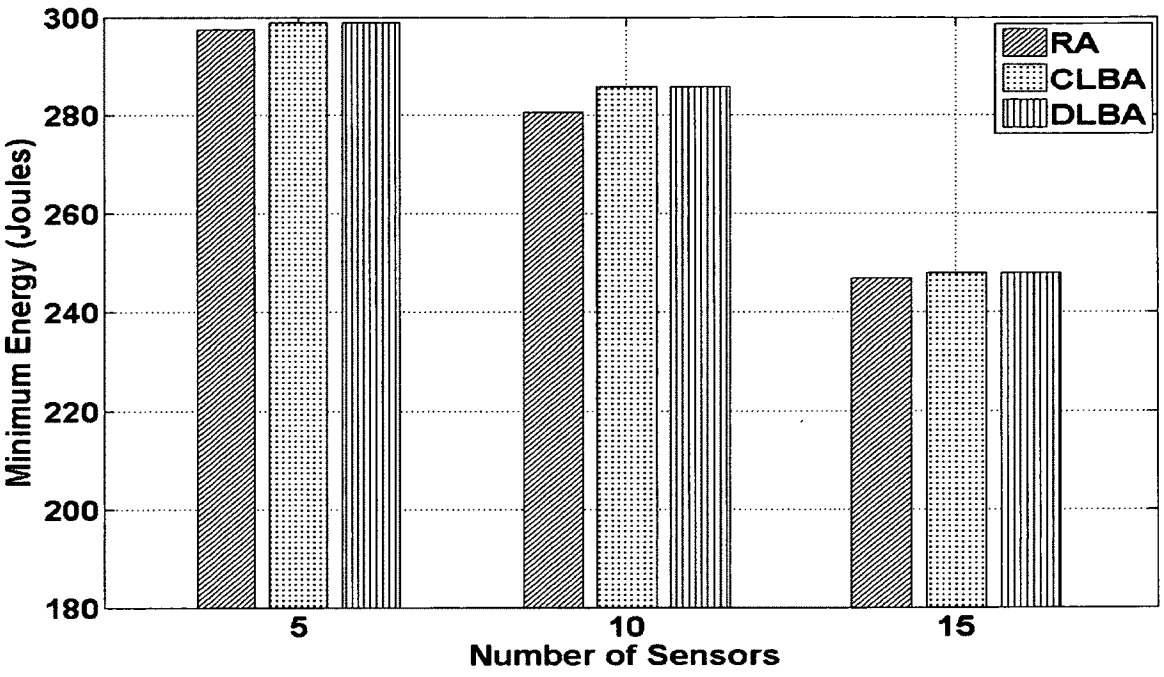


(b)

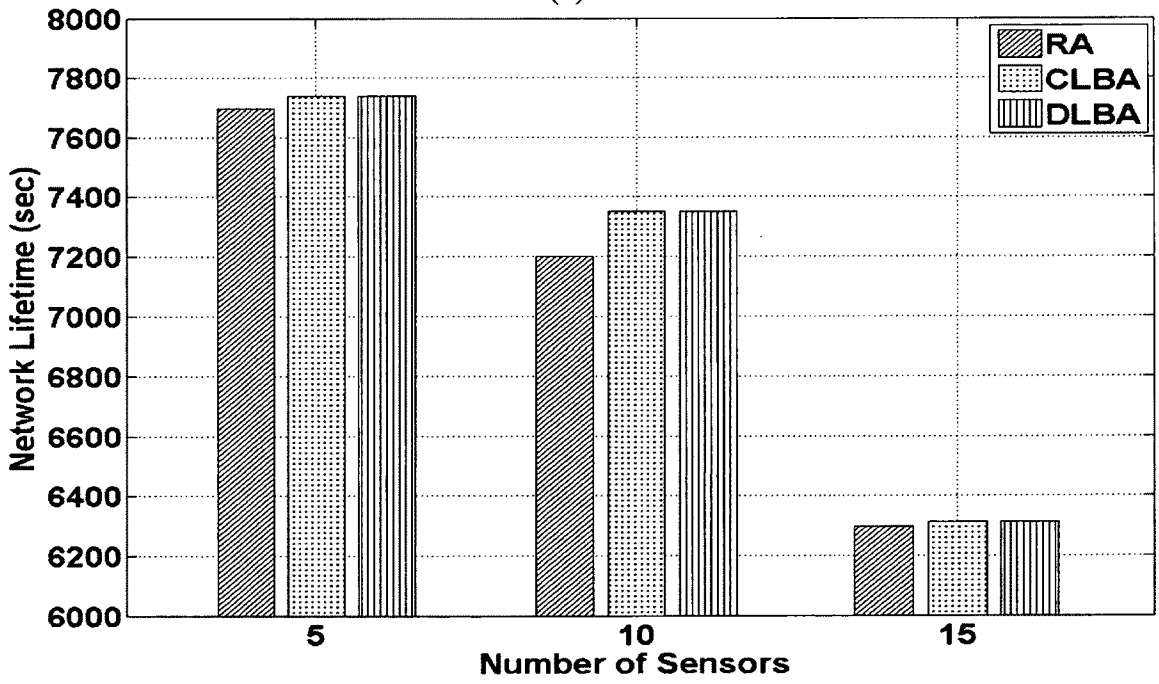
Figure C-4: Effect of Data Size on the Performance of Dynamic Allocation Algorithms with Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

C.5: Effect of Number of Sensors on the Performance of Static Allocation Algorithms



(a)

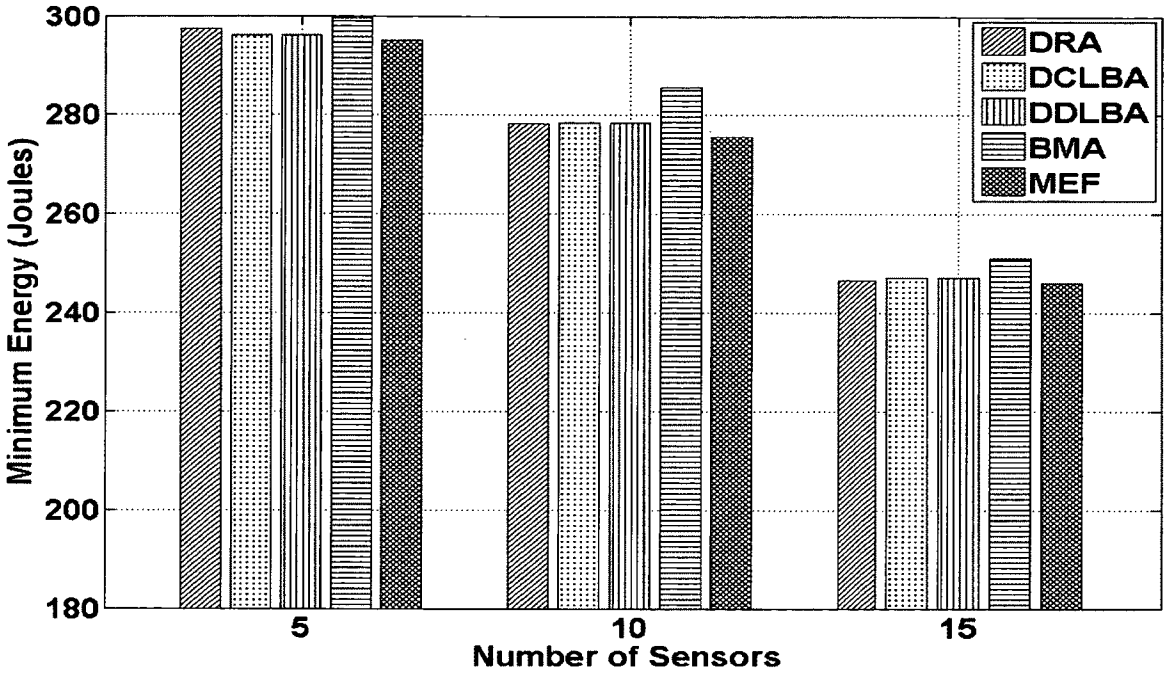


(b)

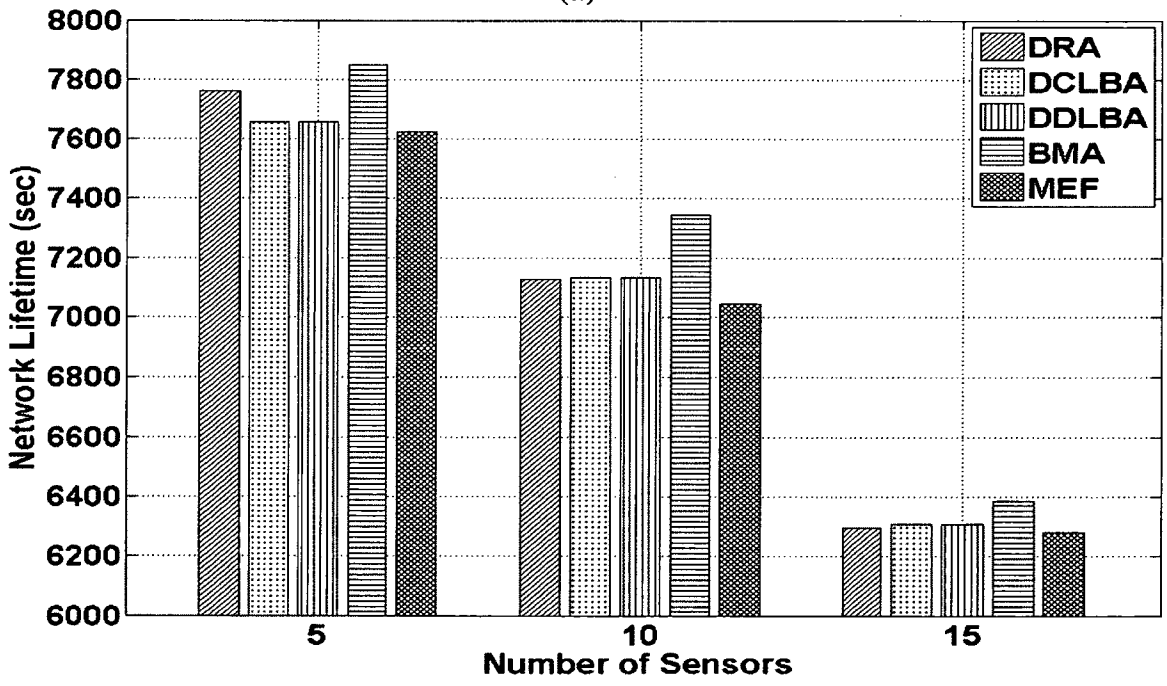
Figure C-5: Effect of Number of Sensors on the Performance of Static Allocation Algorithms with Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

C.6: Effect of Number of Sensors on the Performance of Dynamic Allocation Algorithms



(a)

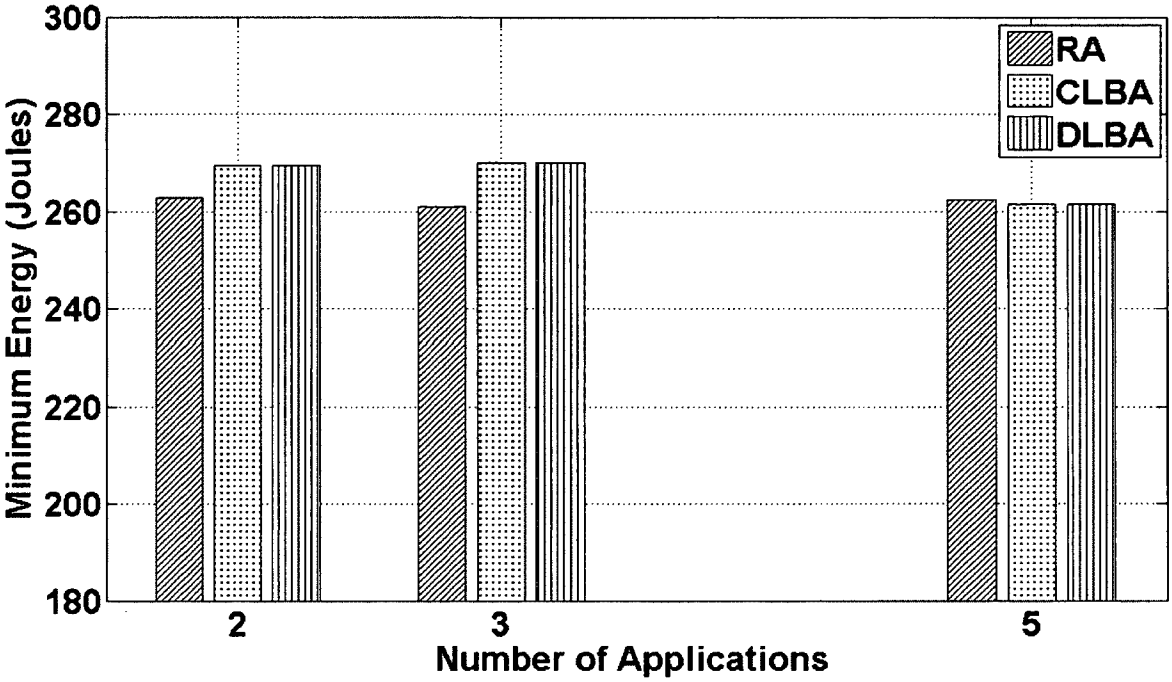


(b)

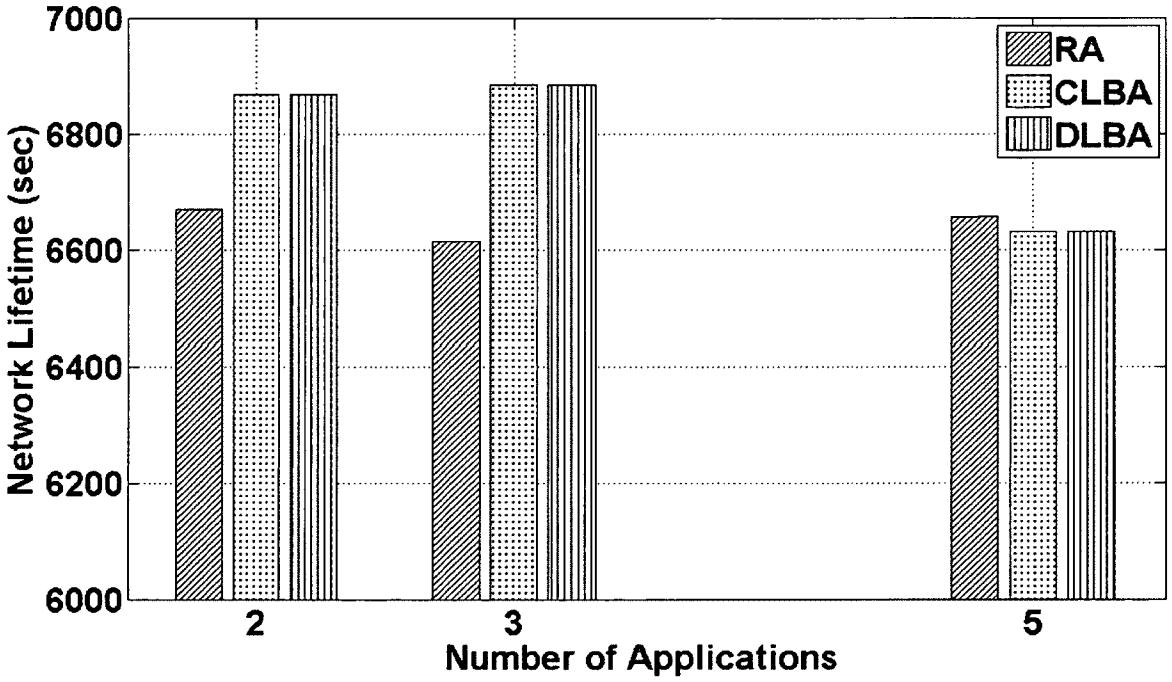
Figure C-6: Effect of Number of Sensors on the Performance of Dynamic Allocation Algorithms with Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

C.7: Effect of Number of Applications on the Performance of Static Allocation Algorithms



(a)



(b)

Figure C-7: Effect of Number of Applications on the Performance of Static Allocation Algorithms with Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime

C.8: Effect of Number of Applications on the Performance of Dynamic Allocation Algorithms

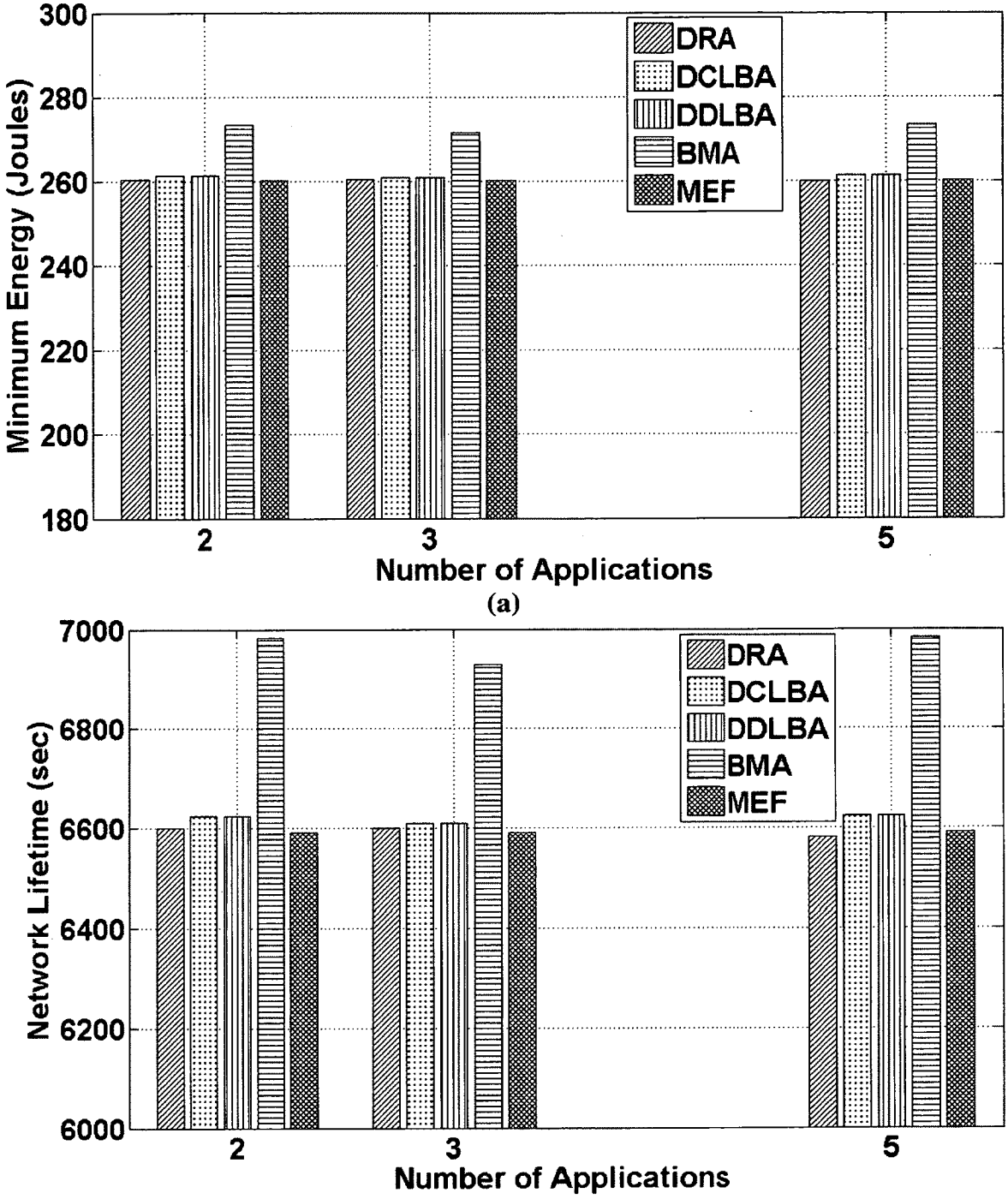


Figure C-8: Effect of Number of Applications on the Performance of Dynamic Allocation Algorithms with Unequal Initial Energy of Sensor Nodes

(a) Minimum Energy at a Sensor Node (b) Network Lifetime