# ENERGY- AWARE TASK SCHEDULING OVER MOBILE AD HOC NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALI BOKAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JANUARY 2009

Approval of the thesis

**ENERGY- AWARE TASK SCHEDULING OVER MOBILE AD HOC NETWORKS**

Submitted by **ALI BOKAR** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Science**

Prof. Dr. Müslim Bozyigit _____
Head of Department, **Computer Engineering**

Prof. Dr. Müslim Bozyigit _____
Supervisor, Computer **Engineering Dept., METU**

Dr. Cevat Sener _____
Co-Supervisor, Computer **Engineering Dept., METU**

**Examining Committee Members:**

Prof. Dr. Nazife Baykal _____
Informatics Institute. METU

Prof. Dr. Müslim Bozyigit _____
Computer Engineering Dept., METU

Assoc. Prof. Dr. Ali Dogru _____
Computer Engineering Dept., METU

Asst. Prof. Dr. Tolga Can _____
Computer Engineering Dept., METU

Asst. Prof. Dr. Ibrahim Korpeoglu _____
Computer Engineering Dept., Bilkent University

**Date: 22 JANUARY 2009**

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last name: **ALI BOKAR**

Signature:

## ABSTRACT

# ENERGY- AWARE TASK SCHEDULING OVER MOBILE AD HOC NETWORKS

**BOKAR, Ali**

**Ph.D., Department of Computer Engineering**

**Supervisor: Prof. Dr. Müslim Bozyigit**

**Co-Supervisor, Dr. Cevat Sener**

**January 2009, 110 pages**

Mobile ad hoc networks (MANETs) can be formed dynamically without the support of any existing infrastructure or any centralized administration. They consist of heterogeneous mobile nodes which are powered by batteries, move arbitrarily and are connected by wireless links. Battery energy limitation is one of the main challenges in the MANETs. Several hardware and software based techniques have been proposed in this field. Most of the previous studies have considered only the energy minimization of individual nodes and disregarded the overall network lifetime. Topology management is another important problem in MANETs, in this sense; several new computing paradigms have been developed by the researchers, and the topology management has not been studied clearly in most of these models.

In this study, we propose two new techniques that deal with the topology management in order to facilitate the nodes' cooperation towards energy saving. The developed computing model considers heterogeneous mobile nodes. A node that faces shortage in its resources (energy and processing capability) sends its

work to one of the nearby devices which is able to execute the work. In addition, we propose two algorithm for dynamic and two for static task scheduling, to prolong the network life time.

Comprehensive experiments showed that the proposed schemes achieve a significant improvement in the network lifetime while simultaneously reducing the energy consumption and time delay for each task.

**Keywords:** Energy Aware Clustering, Energy Aware Design, MANET, Mobile Computing, Task Scheduling.

# ÖZ

## ENERGY- AWARE TASK SCHEDULING OVER MOBILE AD HOC NETWORKS

**BOKAR, Ali**

**Doktora, Bilgisayar Mühendisligi Bölümü**

**Danışman: Prof. Dr. Müslim Bozyigit**

**Ortak Danışman: Dr. Cevat Şener**

**Ocak 2009, 110 sayfa**

Hazırda var olan bir altyapıdan veya merkezi bir yönetimden destek almaksızın dinamik olarak oluşturulabilen ağlara "mobil tasarısız ağ" anlamında MANET denilir. MANET ağlar, farklı cinslerden mobil cihazlardan oluşmakta olup, rastgele hareket ederler, enerjilerini pillerden alırlar ve kablosuz bağlantılara sahiptirler. Pillere bağlı enerji sınırlaması, MANET'lerdeki en büyük sorunlardan birisidir ve bu alanda bir çok donanımsal ve yazılımsal teknik öne sürülmüş bulunmaktadır. Geçmişteki bu tip çalışmaların bir çoğu, tekil cihazların harcadıkları enerjinin azaltılması ile ilgilenmiş olup, toplam ağın yaşam süresi üzerinde durmamıştır. Enerji kazanımına yönelik, uygun topoloji yönetimi, MANET'lerdeki önemli sorunlardan biridir; ancak araştırmacılar tarafından bu alana ilişkin geliştirilmiş yeni paradigmaların çoğunda topoloji yönetimi yeterince açık olarak çalışılmamıştır. Bu çalışmada topoloji yönetimi ile ilgili olarak ağ içindeki cihazların işbirliğini destekleyen iki yeni farklı yöntem önerilmektedir. Burada kullanılan model farklı cinslerden cihazlar içermekte olup, kaynak sıkıntısı çeken bir cihaz, elindeki işi, o işi gerçekleştirebilecek olan yakınındaki bir cihaza göndermektedir.

Ayrıca, önerilen topoloji yönetim yöntemleri üzerine oturtulan ağın yaşam süresini artırmaya yönelik enerji bilinçli dinamik ve statik görev zamanı planlama yöntemleri de önerilmiştir. Kapsamlı deneyler, önerilen yöntemler ile, enerji tüketim değerleri ve bekleme süreleri azaltılırken, aynı zamanda ağın yaşam süresinin de önemli ölçüde artırıldığını gösterilmektedir.

**Anahtar Kelimeler** : Enerji Odaklı Öbekleme, Enerji Odaklı Tasarım, MANET, Mobil İşlem, iş Planı

# ACKNOWLEDGMENTS

First and foremost, I thank God.

My greatest gratitude goes to my supervisor Prof. Dr. Müslim Bozyigit for his guidance and consistent support. His knowledgeable, wise and inspiring discussions have guided me throughout my whole Ph.D. career.

I sincerely thank my co-supervisor, Dr. Cevat Sener for his advice, guidance, patience, and direction.

I would also like to acknowledge the support given by Prof. Dr. Nazife Baykal, as well as Assoc. Prof. Dr. Ali Dogru as members of my thesis committee from the early stage of this study.

I would also like to acknowledge the support given by the staff of Computer Engineering Department throughout my study.

I owe my parents any success I have achieved in my life. Their advices and encouragement motivate and lead me through all my life. I would like to thank my wife Ashjan and our sons Shatha and Amr for their love, constant support, and patience.

My special thanks go to my friend Ayman Elwali who has helped me in correcting the writing of some chapters of my thesis.

I could not forget my friends Ali Binhadjah, Adnan Al-asbhi, and Khaled Banafa with whom I have spent memorable time in Turkey.

Finally I would like acknowledge Hadhramout University of Science and Technology and TUBITAK for their financial support during my study.

To my Parents

To my wonderful wife and our two precious ones- Shatha, and Amr

TABLE OF CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# LIST OF ABBREVIATIONS

AODV Ad hoc On-Demand Distance Vector

AVA(t) average execution time

CC Channel Capacity

$C_{in}$ Intern cluster

CN Client Nodes

CSMA/CA Multiple Access with Collision Avoidance

CS Clustering Scheme

CTS Clear To Send

$C_{upp}$ Upper parent cluster

DAG Directed Acyclic Graph.

DARPA Defense Advanced Research Projects Agency

DCF Distributed Coordination Function

DS Dominating-Set-based clustering

DSDV Destination-Sequenced Distance-Vector

DSS Dynamic Server Selection

DSR Dynamic Source Routing

DVS Dynamic Voltage Scalable

$E_i$ energy level of node i

EADTA Energy Aware Dynamic Task Allocation

EADSSTA Energy Aware Dynamic Server Selection based Task Allocation

EA- STS-DSS Energy Aware Static Task Scheduling using DSS

EA-STS–CS Energy Aware Static Task Scheduling using CS

ETS Energy aware task scheduling

$FH_{tim}$ finishing time

$FH_{max}$ maximum finishing time

FSR Fisheye State Routing

GPS Global Positioning System

IETF Internet Engineering Task Force.

LANMARK Landmark Ad hoc Routing

LAR Location Aware Routings

LS List Scheduling

L(t) Task T level

MAC Medium Access Control

MANETs Mobile ad hoc networks

MFR Most Forward within Radius

MPRs Multipoint Relays

MRL Message Retransmission List

N Number of nodes

*Neig(i)* neighbors of node i

PDA Personal Digital Assistants

PAN Personal Area Network

PCF Point Coordination Function

$P_{in}$ Intern parents

| | |
|---|---|
| $P_{upp}$ | Upper Parent |
| QoS | Quality of Service |
| OLSR | Optimized Link State Routing |
| RE | minimum residual energy |
| $RE_{max}$ | maximum residual energy |
| RTS | Request To Send |
| $S_{ik}$ | node i is a server at level k |
| SEADTA | Scalable Energy Aware Dynamic Task Allocation |
| STAR | Source Tree Adaptive Routing Protocol |
| TCP | Transmission Control protocol |
| TBRPF | Topology Broadcast based on Reverse Path Forwarding |
| TORA | Temporally Ordered Routing Algorithm |
| TR | Transmission Range |
| ZRP | Zone Routing Protocol |
| WiFi | IEEE 802.11 |
| WRP | Wireless Routing Protocol |
| $\rightarrow$ | Parenthood relationship |
| —— | Siblinghood relation |
| $\lambda$ | Task arrival rate |
| $\Rightarrow$ | Single hop accessibility |
| $\Leftrightarrow$ | Symmetric accessibility |
| $\overset{p}{\Leftrightarrow}$ | Multi-hop accessibility |

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction

The idea of ad hoc networking goes back to the U.S. Defense Advanced Research Projects Agency (DARPA) packet radio network, which was used in the 1970s [24]. Mobile ad hoc networks (MANETs) are those which can be formed dynamically without the support of any existing infrastructure or any centralized administration. They consist of various mobile nodes which move arbitrarily and are connected by wireless links. MANET is an attractive technology, due to its easy installation in the areas with no pre-existing communication infrastructure, or when installing such infrastructure is impossible, or when a wireless extension is needed. Moreover, mobile users can get the internet services, if at least one node can connect to a fixed backbone network through a dedicated gateway device enabling IP network services [3].

A MANET group has been formed within the Internet Engineering Task Force (IETF). The main goal of this group is to develop MANET standardizations and introduce them to the Internet standard track. The aim is to integrate mobile ad hoc networks with hundreds of routers and solve challenges in this kind of networks. Some challenges of MANET are battery constraints, mobility induced route change, and packet losses due to transmission [24].

With the advancement of technologies, mobile devices get smaller, more convenient, and more powerful, thus many applications are adapted to run on MANETs. For example, two of its common application areas are disaster recovery management and military applications, where infrastructure networks are almost impossible to form or maintain. Other important MANET applications are dominant in business, for example, salesmen and marketers who travel with

laptops, oil rigs, earth quake detection/prediction, peer-to-peer communication, industrial, crops monitoring, social event monitoring, and others [1,2,3].

Another application example of MANET is Bluetooth, which is designed to support a personal area network (PAN) by avoiding the need for wires between various devices, such as printers and personal digital assistants. The well known IEEE 802.11 also known as WiFi protocol supports MANET system in the absence of a wireless access point [24].

## 1.2 Motivation

Each node in MANET communicates directly to the nodes that stand within its transmission range, however, to route information to the nodes that reside outside its range, it needs to use the intermediate nodes. This technique is called multi-hope routing, and therefore, MANETS sometimes are called multi-hope wireless networks. Multi-hop routing technique brings many problems; for example, by using this technique, each node consumes its energy not only on its benefit work, but also on other nodes' work. Since nodes are battery powered, the probability that some nodes run out of energy will quickly increase. Moreover, multi-hope technique implies that each node acts as a router, which means that there are multiple routers available at any time; and as a result the routing management will be complicated. Other problems of multi-hop are related to security issue.

MANET is infrastructure-less in its nature where each node acts as an independent router and produces independent data. Hence, the network model is fully distributed, and as a result, nodes work in a peer to peer model. Network management is distributed over all nodes, which complicates not only detection of faults but also the network management itself.

Due to the arbitrary movement of nodes in MANET, the topology changes frequently and unpredictably, which leads to the change in multi-hop routing, packet losses, and network partitions.

MANET network consists of diverse wireless mobile devices such as, Personal Digital Assistants (PDA), mobile phones, sensor platforms, and laptops. These devices are different in their processing capabilities and are not only equipped with different radio interfaces, but also some of them have more than one radio interfaces which are working at different frequency. As a result, this may lead to asymmetric links. Consequently, designing network protocols and algorithms for such heterogeneous networks is more complex.

Wireless Mobile Devices depend on the batteries; such batteries are limited in power supply, which results in limitation in services and application that can be provided by each node. Moreover, each node does not consume its energy only for its own work, but also transfers messages on behalf of other nodes. Therefore, the issue of energy conversion is bigger in MANET than other networks model.

Some MANET applications require thousand of nodes. The successful deployment of such large number of heterogeneous nodes is not easy and requires many aspects such as, addressing, routing, location management, security, and others [3].

One solution of the heterogeneity in node capabilities and energy limitation is to let nodes cooperate to benefit from the resources on all nodes in the network. Therefore, in this study we design a cooperative energy aware task scheduling in which nodes that are unable to execute a task, due to processing incapability or energy limitation can benefit from the resources on other nodes.

## 1.3 Problem Statement and Contributions

Energy constrain and dynamic topology are two main challenges of MANET. Therefore, many researchers studied these two problems in different protocol layers, for example, studies in [4,5] introduce two power aware routing protocols, on the other hand, authors in [6,7,8] study power control on MAC level.

In this study, we tackle the above problems from the perspective of task scheduling. The main objective of this study is to design a cooperative model for energy aware task scheduling over mobile ad hoc networks which maximizes the overall network life time and facilitates the topology management. To best of our knowledge, this is the first study that manipulates these two issues together. The main contributions of this study are:

a- Two new innovative techniques are introduced to cope with topology management in order to reduce the required energy for nodes cooperation and to orient remote loads to the nodes that have the highest energy. In the model, the network lifetime is divided into equal time rounds, at the beginning of each round; one of the techniques listed below is executed.

1- Dynamic Server Selection (DSS)

The goal of DSS is to classify the network nodes into servers and non-server nodes. In addition, the servers themselves are also classified into servers at level one and servers at level two according the explained criteria in chapter 3. These servers will be used in the remote task allocation based on the proposed technique.

2- Clustering Scheme (CS)

The aim of CS is to structure the network nodes into parent nodes and non-parent nodes. In fact, the parent nodes are divided into three types of parents: the upper parents, the three hop parents and the parents. The structure of the network nodes is that the upper parents are the parents of three hope parents, whereas the three hope

parents are the parents of the parents, on the other hand, the parents are parents of the non-parent nodes. This classification is based on the energy level of each node.

b- In addition to DSS and CS, we propose two new dynamic energy aware task allocations. The Energy Aware Dynamic Server Selection based Task Allocation (EADSSTA), and Scalable Energy Aware Dynamic Task Allocation (SEADTA). While EADSSTA is implemented over DSS, SEADTA is implemented over CS.

c- Energy Aware Static Task Scheduling using DSS (EA- STS-DSS) and Energy Aware Static Task Scheduling using CS (EA- STS –CS) are two new energy aware static tasks scheduling which are designed to execute over DSS and CS respectively.

### 1.4 Thesis Outline

The remainder of this document is organized as follows. Chapter 2 gives an overview over mobile ad hoc networks including challenges, network layer protocols, and MAC layer. Traditional task scheduling is also reviewed in this chapter. Chapter 3 is to clarify the aspects of both DSS and CS. Chapter 4 explains the energy aware dynamic task allocations, and it specifically presents the details of EADSSTA and SEADTA schemes. Chapter 5 details the energy aware static scheduling EA- STS-DSS and EA- STS –CS. In chapter 6, we present a comprehensive simulation study of the proposed techniques. We conclude in Chapter 7, and show some directions of the future work.

# CHAPTER 2

# BACKGROUND

## 2.1 Mobile Ad Hoc Networks

In wireless mobile ad hoc networks, mobile nodes operate without the help of an established infrastructure of centralized administration. Communication is done through wireless links among nodes that exist in transmission of each other; however, a mobile node is unable to communicate directly in one hope fashion with the nodes that stand outside its transmission range, therefore, a multi-hope scenario occurs, where the packets sent by the source node are relayed by several intermediate nodes before reaching the destination node [25].

The primary application of MANET were military related, however, with the rapid advances in MANET researches and the introduction of a new technologies such as Bluetooth and IEEE 802.11, new applications in different domain areas have been developed, for example, application for emergency services, disaster recovery, and environmental monitoring. Table 2.1 shows some important MANET applications in different areas.

Table 2.1: MANETs applications (taken from [14])

| Application | Descriptions/Services |
|---|---|
| Tactical Networks | • Military communication, operations<br>• Automated Battlefields |
| Sensor Networks | • Home applications: smart sensor nodes and actuators can be buried in Appliances to allow end users to manage home devices locally and remotely<br>• Environmental applications include tracking the movements of animals (e.g., birds and insects), chemical/biological detection, precision agriculture, etc.<br>• Tracking data highly correlated in time and space, e.g., remote sensors for weather, earth activities |
| Emergency Services | • Search and rescue operations, as well as disaster recovery; e.g., early retrieval and transmission of patient data (record, status, diagnosis) from/to the hospital<br>• Replacement of a fixed infrastructure in case of earthquakes, hurricanes, fire etc. |
| Commercial Environments | • E-Commerce: e.g., Electronic payments from anywhere (i.e., taxi)<br>• Business:<br>- dynamic access to customer files stored in a central location on the fly<br>- provide consistent databases for all agents<br>- mobile office<br>• Vehicular Services:<br>- transmission of news, road condition, weather, music<br>- local ad hoc network with nearby vehicles for road/accident guidance |
| Home and Enterprise Networking | • Home/Office Wireless Networking (WLAN) e.g., shared whiteboard application; use PDA to print anywhere; trade shows<br>• Personal Area Network (PAN) |
| Educational applications | • Setup virtual classrooms or conference rooms<br>• Setup ad hoc communication during conferences, meetings, or lectures |
| Entertainment | • Multi-user games<br>• Robotic pets<br>• Outdoor Internet access |
| Location aware services | • Follow-on services, e.g., automatic call-forwarding, transmission of the actual workspace to the current location<br>• Information services<br>- push, e.g., advertise location specific service, like gas stations<br>- pull, e.g., location dependent travel guide; services (printer, fax, phone, server, gas stations) availability information |

### 2.1.1 MANET Issues

Due to its dynamic topology change, wireless links, and multi-hop routing, MANET brings many new issues to the computer network design. These issues are currently hot research areas. In this section, we present some of the problems related to MANET.

- Transmission Control protocol (TCP) is a connection-oriented transport control protocol which provides the essential flow control and congestion control required to ensure reliable packet delivery. The multi-hop mobile networks introduce new challenges to TCP protocol, such as impact of mobility and TCP congestion window size, and nodes interaction MAC layer.

- The mobility of nodes leads to the frequent and unpredictable change in the network topology; consequently, routing management is difficult and complex in such networks.

- Vulnerability of channels and nodes, absence of infrastructure, and dynamically changing topology make ad hoc networks' security a difficult task, in addition that the broadcast wireless channels allow message eavesdropping and injection. Due to the absence of the infrastructure, the classic security methods are inapplicable in MANET [24].

- The characteristic of all the network components specify the type of Quality of Service (QoS) that the network can support. Since Wireless links have a low and highly variable capacity, and packet loss, in addition that topology is highly dynamic change, QoS in MANET needs many problems in different layers to be addressed.

## 2.2 The IEEE 802.11 (WiFi)

The development of IEEE 802.11 facilitates the deployment of wireless LAN. You can see wireless networks in many places such as cafes, airports, and educational institutions. The 802.11 specifies two modes of MAC protocol, the Distributed Coordination Function (DCF) mode (for ad hoc networks) and Point Coordination Function (PCF) mode (for centrally coordinated infrastructure-based networks). There are many 802.11 standards for wireless LAN technology; the important three ones are 802.11b, 802.11a, and 802.11g. they share many characteristics. They all use the same medium access protocols Carrier Sense Multiple Access with Collision Avoidance CSMA/CA. each station senses the channel before transmitting, and refrains from transmitting when the channel is sensed busy. In addition, they use Request To Send RTS and Clear To Send CTS reservation scheme that helps avoid collisions, also they use link layer acknowledgement scheme for each frame. Moreover, all three are able to reduce their transmission rate in order to enlarge their recover distance. Although they are shared in many features, they are different in their physical layer (data rate and frequency rate) [24, 71].

## 2.3 Routing in Mobile Ad Hoc Networks

The frequency and unpredictability of node mobility in MANET lead to the dynamic topology, therefore, traditional routing protocols, which are designed to work on the wired and fixed network, are inapplicable for the MANET. In literature, numerous MANET routing protocols have been developed and their performance also has been studied under different conditions [1].

Routing protocols can be classified in several ways, for example, according to the way by which they forward the messages, i.e. weather they use unicast, geocast, multicast, or broadcast. Another way of classification is to divide the routing protocols to topology-based and position-based approaches

[3]. Topology-based routings are further divided into proactive, reactive, and hybrid protocols.

### 2.3.1 Proactive Routing protocols

These protocols are developed based on the mechanisms used by the traditional routing protocols, distance-vector and link-state protocols to keep track updated routing information to any node in the network. to get these updated information, each node propagates routing information message in the network continuously during a fixed time interval. Since routing information is updated regularly and maintained in routing tables, these protocols sometimes are called table-driven-protocols. Proactive protocols show minimum delay when the routing information is needed, however, they consume high energy and decrease the network capacity, due to the updating routing table continuously.

The Destination-Sequenced Distance-Vector (DSDV) protocol [19] is an extension of the traditional distance-vector protocol in order to make it suitable for MANET. Each node keeps a routing table includes one entry for each possible destination with the shortest path (based on the number of hops) to reach it. Moreover, each entry is assigned with the sequence number generated by the destination node, in addition, each node increment its sequence number whenever a change occurs in its neighborhood. To avoid loops, each node selects the router with the highest sequence number.

Another proactive protocol is the Optimized Link state Routing (OLSR) [20] , which is developed based on the legacy link state protocols. OLSR reduces the size of control packets by broadcasting only subset of the neighboring links. These subset nodes are called Multipoint Relays (MPRs). Each node selects some nodes (from its one-hop neighbor) that cover all its two-hop nodes to work as it MPR, which is updated over the time. The Topology Broadcast based on Reverse Path Forwarding (TBRPF) [21] protocol is an another example of the link state routing protocols which reduces the overhead of control packets by

making each node computes its shortest path tree to nodes in the network, however, to save the bandwidth , the node broadcasts only part of its tree to its neighbor nodes.

The Source Tree Adaptive Routing Protocol (STAR) [22] allows the route to be non-optimal to save the bandwidth, therefore, it does not use periodic message to update the routing table, and instead, it depends on the underlying protocols to maintain the neighboring nodes. The reliable broadcasting is required to implement STAR. This service is either provided by the link layer or it is built within the STAR itself. The wireless Routing Protocol (WRP) [23] is a table driven protocol in which each node in the network keeps four tables: Distance table, Routing table, Link-cost table, and the Message retransmission List (MRL) table. Each node gets the changing information of all links by exchanging update messages among all neighbors' nodes. WRP breaks the routing loop by using the shortest path to each destination.

### 2.3.2 Reactive Routing Protocols

Contrast to the proactive routing protocols, reactive routing protocols build the routing to the destination only when it is needed. The node enters to the route discovery process by initiating the route request message in the network, once the route has been built, the nodes becomes ready to send packets. Moreover, the node keeps route information until either the destination is no longer accessible, or until the route is expired. The main advantage of the reactive protocols is that they use much less bandwidth than the proactive protocols; however, they show high delay to the routing request.

The Dynamic Source Routing (DSR) [24] forwards packets from the source to the destination based on the routing information carried by each data packet. The routing information is injected into the header of each data packet by the source node. Each node keeps route caches which consist of source routes that the node has learned. The route caches are updated when new route

information receives the node. When a node has a packet to send, first, it checks its route caches to see whether it has valid route information to the destination. Route discovery process is initiated only if the node does not have valid route information.

The Ad hoc On-Demand Distance Vector (AODV) [26] is an improvement of both DSDV and DSR. It minimizes the number of route broadcasts by creating routes on-demand similar to DSR; moreover, unlike DSDV which keeps the complete list route, AODV keeps only part of the complete list route.

The Temporally Ordered Routing Algorithm (TORA) [27] is one of the reactive on-demand routing protocols which is built based on the concept of link reversal. It is designed to minimize the effect of the topology change when a change happens in some place. The control message that reflect this change is not sent to all nodes, instead, it is sent to only small set of nodes near the change. TORA provides multiple paths from source/destination pair, this property makes it appropriate for the environment with large number of nodes and highly dynamic change.

The ABR [23] is another reactive routing protocol which depends on a new metric to select a router. This metric is called degree of association stability, the idea is that ABR tries to select the long-live router which is stable and does not need more updates subsequently. The main drawback of ABR is that it uses periodic messages to build the association stability metrics.

### 2.3.3 Hybrid Routing Protocols

Hybrid routing protocols try to bring the advantages of both proactive and reactive protocols together. They use the proactive methods to send routing information to the close nodes; on the other hand, they use reactive technique to get routing information to the distance nodes.

The Zone Routing Protocol (ZRP) [24] defines a zone area around each node based on some selected number of hops. ZRP uses proactive mechanism to forward message inside zone and reactive techniques to forward messages outside the zone.

The Fisheye State Routing (FSR) [23] is an optimization link-state algorithm which distributes the link state information in the network based on the scope term which is defined by the number of hops. The protocol exchange link-state information frequently in the closer scope, however, it exchanges information less frequent to the nodes that stand further away.

Landmark Ad hoc Routing (LANMARK) [1] divides the network nodes into different mobility groups, one called landmark is selected in each group to maintain in which group each node is a member of, and coordinate inter-group routing, on the other hand, FSR is used for intra-group routing.

### 2.3.4 Location-Aware Routing (LAR)

Location Aware Routings (LAR) come to avoid the routing information request messages used by proactive, reactive, and hybrid protocols. LAR depend on the availability of node information location services, for example, Global Positioning System (GPS). Usually the sender node uses the location service to get the location information of the destination node; moreover, the sender adds the destination location information to each packet. Depend on the location information of both destination and its one hop neighbors, the sender node selects the next one hope forwarding node. The intermediate receiver nodes use the destination location information included in the packet and the physical location of their one hope neighbors to compute the next one hope forwarding node. The disadvantages of LARs are that they depend on external services and assume that each node has additional hardware equipment.

LARs use three methods for forwarding packets: greedy forwarding, directed flooding, and hierarchical routing [23 ]. In greedy forwarding algorithm, nodes try to forward packets to one of their neighbor that is the closest to the destination node. One of the mechanisms that is used when more than one closer exist is the Most Forward within Radius (MFR) policy which forwards packets to the node closest to the destination. The nearest with Forward Progress (NFP) sends the packets to the node that closer to the sender, another method is the compass routing scheme which forwards packets to the neighbor that is closer to the straight line joining the sender to the receiver. While GPRS and the Face algorithm [23] use MFR scheme for selecting the next node, The Geographical Distance Routing (GEDR) [23] uses the MFR and the compass routing scheme.

In Directed Flooding, nodes forward the packets to all neighbors that are located in the direction of the destination. DREAM and LAR [1] are two algorithms that use this technique. In Hierarchical routing, routing is structure in hierarchical layers in order to scale the network. Terminode routing [1] and Grid routing [23] protocols are two types of the hierarchical routing in which routing is achieved by using proactive distance vectors scheme, however, for the long distance node, they use a greedy position based approach.

## 2.4 Energy Awareness Related Work

Mobile nodes depend on the carried batteries in their work. These batteries are limited in power supply; therefore, many researches try to develop techniques that aim to prolong the battery life. In this concern, several hardware and software methods have been introduced.

### 2.4.1 Hardware Techniques

Many hardware components of mobile node consume the energy. One solution to decrease the energy consumption is to design these devices to work

with low energy. In general, two hardware techniques can be used for this sake; the first is to have an automatic switch off for the idle device in order to save energy, whereas the second method works in the Dynamic Voltage Scalable (DVS) processors. DVS processors have different speed levels which can be configured by changing the supply voltage. Based on the application requirements, the processor's speed is manipulated accordingly [28].

In [29, 30, 31, 32] the authors Study the variable voltage scheduling of DSV processors to minimize the energy consumption by adjusting the voltage and frequency of the processor, on the other hand, studies in [33, 34] present a either a hardware turned off, or decrease the load in components that have low energy.

Some other studies combine between software methods and hardware techniques, for example, Study in [35] proposes five methods to decrease the energy consumption of clustering web servers that use DSV processor. According to the load, the web servers are turned off or work in low voltage. While study [36] presents a method to shut down the devices when they are waiting for events (blocked state), such as waiting for input data or user to press a key, the authors in [31,37] Describe software approach to disk driver spin down/spin up to decrease the energy consumption by the disk. The approach in [54] orders task execution such that different components of a mobile computing device can have longer idle periods to be shut down.

### 2.4.2 Software Techniques

Many efficient software protocols have been designed to work in different layers in order to reduce the energy consumption. Power aware routings have been proposed in [4, 5] in which power aware metrics are used for determining routes. Study in [53] uses the compiler loop optimization techniques and voltage scaling methods to generate energy aware code. In [56], the authors propose a power-aware message scheduling algorithm for real time system; the basic idea is

that the base station receives real time requests from clients. To conserve energy, the base station tries to schedule replies in a manner that satisfies real time requirement and conserves energy. In [58] the authors introduce an energy-aware adaptation technique, which depend on the dynamic balancing of application quality and energy conservation. However, this method often requires application fidelity to be significantly degraded [59].

#### 2.4.2.1 Energy Aware Task Scheduling Over MANET

One of the recent software methods is the energy-aware task scheduling, in which the tasks are scheduled by taking their energy (execution and communication) into account. In the recent years, a significant amount of work has been done in this field and new computing paradigms have been developed by researchers. In this section, we classify the energy aware tasks scheduling as follows: energy aware task scheduling using DVS processors, energy aware task scheduling inside mobile clustering, Computation offloading (remote execution), and cooperative computing model. The following sections discus these models.

##### 2.4.2.1.1 Energy Aware Task Scheduling Using DVS Processors

In [28,55], the authors propose a two-phase real time dynamic task scheduling algorithm for the battery operated DVS system in order to maximize both, the residual energy and the battery voltage. This proposal is valid for single processor systems, on the other hand, study in [57] Proposes low power real time system-on-chip task scheduling based on dynamically variable voltage, moreover, it addresses the selection of the processor core and the determination of the instruction and data cache size and configuration so as to fully exploit dynamically variable voltage hardware. Study in [60] introduces an energy-aware task allocation scheme for parallel applications on heterogeneous embedded systems whereas the authors in [61] propose an energy-balanced allocation of a

real-time application onto a single-hop cluster of homogeneous sensor nodes connected with multiple wireless channels, and each sensor node is equipped with discrete dynamic voltage scaling (DVS).

Aperiodic and periodic static task scheduling for a single and a multiprocessor DVS system was proposed by [62], this study introduces a heuristic algorithm based on the distribution of the available task's slack, in order to decrease the cost function. In the same line, the study in [63] focuses on/considers two problems, the minimizing scheduling length with energy consumption constrain and the minimizing energy consumption with scheduling length on DVS multiprocessor computers. It presents an analytical comparison on the performance, proposes optimal solutions, and verifies the results experimentally.

#### 2.4.2.1.2 Energy Aware Task Scheduling in Mobile Clustering

In [64] the authors define the mobile cluster as a collection of mobile and non-mobile nodes, which can communicate via wireless networks or high performance networks in order to collaborate in their work. This study also makes a general explanation of the hardware and software components and some applications of mobile clusters. On the other hand, the study in [65], proposes a mobile clusters which consists of mobile and stationary nodes. The nodes are grouped based on their interest tasks and each group has a coordinator which is responsible for managing the membership of each group. The model in [66] extends the model in [65] by adding the issues related to the mobility.

Study in [2], introduces two energy aware duplication-based task scheduling in the homogenous mobile cluster environment. The objective is to reduce the communication energy. However, although it studies the mobile cluster, cluster formation is not considered in this study.

#### 2.4.2.1.3 Computation offloading (remote execution)

In the computing offloading paradigm, the system consists of two types of nodes, the first are the full power stationary nodes (stations). The second are the mobile devices which are powered by batteries. These mobile devices send their work to one of the full powered station when they face shortage in their recourse (energy). This method is hoped to improve not only the energy consumption, but also the performance [67].

#### 2.4.2.1.4 Cooperative Computing Model

The concept of cooperation is an essential issue in MANETs. Each node relies on other nodes to route their packets to the nodes that stand outside its transmission range. Another type of cooperation is the one in upper layers protocols. Many research groups have introduced the concept of the cooperative computing models, in which the limited-resource device can benefit from the available resources in the nearby devices.

Energy-aware tasks scheduling, in the cooperative computing model, have been studied by many researchers. In [68, 69, 70], the cooperation computing model was studied over the mobile wireless terminals which were connected by a short range. The main idea is to distribute the load (tasks) among the terminals which have a low performance speed by using the DVS techniques. This will, in turn, maintain a considerable save in the overall consumed energy. Moreover, in this technique, the task's time constraint is also satisfied.

Study in [18] presents an energy-aware dynamic task allocation scheme over a cooperative MANET. That is, tasks arrive at each node dynamically. When a task faces processing and/or energy inefficiency at a node, it is transferred to another node in the network such that the total cost is minimized. The scheme tries to minimize both time and energy. Moreover, it allows varying weight of time and energy in the cost function when different applications have

different constraints on time or energy. For example, if an application is more delay constrained, the time metric can be given a higher weight in the cost function, and if an application is more energy constrained, the energy metric can be given a higher weight. The problem is formulated by the following objective function:

$$F(i) = \text{Min } [\alpha 1 * F1(PTj) + \alpha 2 * F2(Qj) + \alpha 3 * F3(Comm[ij]) + \alpha 4 * F4(Ej, Cj) + \alpha 5 * F5(Ecomm[ij])] \text{ for } i, j = 1, 2 \ldots n,\ j \neq i.$$

where $\alpha 1$, $\alpha 2$, $\alpha 3$, $\alpha 4$, $\alpha 5$ are variable coefficients, *F1* is a function of task processing time on node *j*, *F2* is a function of task queuing time at node *j*, *F3* is a function of communication cost of sending a task from node *i* to node *j*, *F4* is a function of battery level on node *j*, and *F5* is a function of energy consumption of sending a task from node *i* to node *j*.

As a node needs to communicate to all nodes in order to perform remote execution, this results in:

a- Consuming a lot of energy by the sender to send a request to all nodes even though the sender uses broadcasting or multicasting techniques.
b- The sender waits long time to receive replies from all nodes.
c- Sending node receives many replies which may lead to overwhelm it.
d- After receiving replies from all nodes, the sender consumes a lot of energy in computing the best node to send it the task.
e- This study does not consider the network lifetime.

The study in [15] formulates the energy-aware scheduling problem and proposes a heuristic algorithm to solve it. The scheduling algorithm schedules a set of computational tasks, which may have dependencies and communication, into a set of heterogeneous processors in such a way that minimizes both the total consumed energy and the makespan (i.e., the time by which all tasks complete their execution). The problem formulated as follows:

Given a task model $T$ and a processor model $P$, find a schedule $S$ that maps each task to a processor and determines the starting time of each task in such a way that minimizes the following cost function:

$$Cost = \lambda\ makespan + \beta\ total\_consumed\_energy$$

- makespan is the time by which all tasks are completed.
- total_consumed_energy is the sum of energy consumed by task execution and inter-processor communication.
- $\lambda$ (token/time unit) and $\beta$ (token/energy unit) are system attributes representing relative importance weights between timing requirements (performance) and energy conservation. $\lambda$ and $\beta$ are also used to map time units and energy units to generic cost unit, namely, *cost tokens*.

The above study only formulates the energy problem and does not study the topology management and the way by which the nodes cooperative. Moreover, the study does not consider the network lifetime.

## 2.5 Clustering in MANETs

A clustering is a technique which is used to divide the mobile nodes into some groups according to some criteria, such as, node energy, connectivity and so on. Clustering is a good method for the dynamic topology management and achieves a better performance in the topology with large number of mobile nodes [38]. Clustering provides many benefits for MANET, for example, nodes inside each cluster can save their transmission energy which results from collision by coordinating their transmission with the help of a specific node called cluster head. Moreover, cluster technique enables the spatial reuse of frequencies, for example, two clusters that are not neighbored can use the same frequency. In addition, clustering facilitates the information routing among nodes by using the cluster head and gateways nodes [9]. Clustering also simplifies the mobility management, for example, if one leaves it cluster and attends to another cluster, only nodes in these two clusters are needed to update their information [39].

Although clustering gives many benefits to MANET, clustering maintaining in dynamic topology is not easy and needs exchanges some messages among mobile nodes, therefore, if the change in topology is high, this may lead to the increase in the messages related to the clustering maintenance which consumes considerable bandwidth and drains the node energy [40]. Another drawback of the clustering is the rebuild of cluster over the whole network. Some clustering schemes require that the cluster has to rebuild overall network when some events take places such as, a movement or a death of a node. This problem is known in the literature as the ripple effects of re-clustering [9]. Some cluster schemes generate clusters with more than two-hops; however, the management of such clusters is complicated, especially, in high mobility environment [38]. Most clustering schemes divide the required time of clustering formation in two phases or more. In addition, they assume that the nodes do not move during the period of formation. Since the mobility of nodes is not known, prolonging the formation period makes some schemes inapplicable [9].

### 2.5.1 Clustering Classification

Clusters can be classified according to many factors, such as; the numbers of hops that separate each pair in a cluster, in this sense, clusters are divided in to one hope, two hops, and multi-hopes clusters. Another factor of classification is whether there exists a special node called cluster head or not. Moreover, the objective of clustering is another way to classify clustering, based on these criteria; clusters are divides into many types, such as, energy aware, mobility based, load balancing and so on. In this section, we use the objective based method to classify the clusters.

Dominating-Set-based clustering (DS) algorithms [41, 42] try to find connected dominating set nodes in MANET. A set is DS if all the nodes in the system are either in the set or neighbors of nodes in the set. DS nodes are used as

routing nodes in order to reduce the number of nodes that maintain the routing table. Such algorithms are proposed and discussed in [9, 39, 41, 42, 43].

Another type of clustering algorithm are the low-maintenance clusters which aim to provide a stable infrastructure for the upper layer protocols by reducing both the re-clustering times and the explicit messages for cluster maintenance, examples of such algorithms are in [40, 44, 45, 46]. In mobility aware clustering algorithms, nodes are divided into groups by putting the nodes that have similar mobility pattern in one group with the hope to improve the stability of the cluster structure against the node mobility [38, 44, 47].

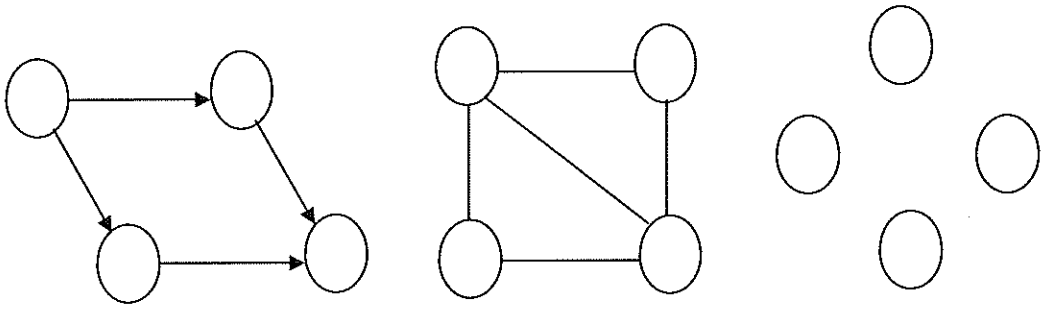
The objective of the energy aware clustering algorithms is to prolong the network life time by avoiding the unnecessary energy consumption and balancing the energy consumption among nodes [9, 48, 49]. Load balancing is another clustering technique which aims to distribute the load among many nodes in the network in order to balance energy consumption [48, 50]. Multilayer cluster is a way by which the network nodes are clustered in hierarchy layers and assigned different jobs to each layer, for example in [51] multilayer clusters are used to facilitate the service discovery in MANET. Some other methods group the nodes based on more than one metric, such as, energy and mobility, in addition, each metric is given a different weight depending on the upper layer application requirements [52].

## 2.6 Scheduling Problem

The scheduling problem, in its general form, can be described by the availability of set of resources and a set of consumers that want to use the recourses according to a certain policy. The primary objective of the scheduling is to find an efficient policy for managing the assignment of the recourses to various consumers to optimize some desired performance measures, such as, scheduling length, resources utilization and so on [11].

In parallel and distributed programming the scheduling problem aims to schedule processes on to multiple processing units in order to improve the performance and efficiency of the system and the application. Scheduling techniques can be divided into local and global methods. Local scheduling is used in single processors to divide the processor time slices among concurrent processes, on the other hand, the global scheduling deals with the assignment of task on to multiple processors [12]. Our attention here is paid to the global scheduling.

The main things that affect the mapping processes into multiple processors are the processes interactions, in this sense, the interactions among processes can be divided into three types, the precedence process model, the communication process model, and the disjoint process model, these models are depicted in Fig 2.1. In the precedence process model as illustrated by Fig 2.1(a), processes are represented by a Directed Acyclic Graph (DAG). While the vertices represent the processes, the edges in the graph indicates the precedence relationships between processes and may show communications overhead if the processes connected by an edge are assigned to different processes.



(a) Precedence Process (b) Communicati on Process (c) Disjoint Process

Figure 2.1: Interaction Process

Fig 2.1(b) shows the communication process model, where processes are created to coexist and communicate asynchronously, and the undirected edges indicate only the need for communication between processes. Since the execution time is not sequenced in the model, the objective of scheduling may be to optimize the total cost of communication and computation. The disjoint process model in Fig 2.1(c) shows that the interaction between processes is implicit, therefore, the processes can be run independently. Processes are mapped to the processors to maximize the processors utilization and minimize the overall completion time.

Scheduling in parallel processing is divided in to two main types, the static scheduling and the dynamic. In static scheduling, which is called also deterministic scheduling, information related to the task precedence is known before the scheduling decision is taken, moreover, the mapping of processes to processors is determined before the execution of the process. When the task graph topology is not known before the program execution, the parallel processor system must schedule the tasks during the execution time, this is known as the dynamic scheduling.

### 2.6.1. Static Scheduling

To formulate the scheduling problem, we need to describe the parallel program tasks (consumers), the target machine (resources), and the cost function [11].

#### Task Model T

Task model is a graphical or mathematical representation that describes all characteristics of the tasks to be scheduled.

In [12] the authors give the following general model of the scheduling problem.

**Task**

$T= [ T_1,..., T_n]$ is a set of task to be executed.

- $<$ is a partial order defined on T which specified the precedence constraints. that is, $T_i<T_j$ means that $T_i$ must be completed before $T_j$ can begin.
- $D_{ij}$ is an nxn matrix of communication data, where $D_{ij} > 0$ is the amount of data required to be transmitted from task i to task j, $1 \leq i, j \leq n$.
- $ET_{ij}$ is an nxm matrix, where m is the number of processor and ET(i,j) is the required execution time when task i is executed on processor j.

**Processor**

- $P=[P_1,..,P_m]$ is a set of processors available on the system.
- CT is an mxm matrix, where CT(i,j) is the amount of time required to send one unite of data from processor i to processor j.

**Cost Function C**

C=Makspan

C= execution_time + communication_time.

Where makspan is the time by which all tasks have completed their execution, execution_time is the total amount of time spent for the execution of all the tasks, and communication_time is the time spent for interaction among tasks that assigned to different processors.

Static process scheduling has been proved in literature to be NP-complete. However, some very restricted cases such as scheduling task with unit time execution on processor model, have known polynomial time as showed in table 2.2 [12]. Therefore, most researches are directed toward sub-optimal (heuristic or

approximate) techniques. A good heuristic algorithm is one that can best balance and overlap computation and communication.

Table 2.2: Complexity comparison of scheduling problem. (taken from [12])

| Task Graph | Task Execution Time | Number of Processors | Complexity |
|---|---|---|---|
| Tree | identical | arbitrary | O(n) |
| arbitrary | identical | 2 | $O(n^2)$ |
| arbitrary | identical | arbitrary | NP-complete |
| arbitrary | 1 or 2 time units | ≥2 | NP-complete |
| arbitrary | arbitrary | arbitrary | NP-complete |

### 2.6.2 Dynamic Scheduling

In the absence of knowledge about the computation and the communication, we have to rely on an ad hoc scheduling strategy that is dynamic and allows its assignment decisions to be made locally. The objectives of the scheduling are the utilization of the system and the fairness to the user processes. To achieve load sharing and balancing in dynamic scheduling, tow process transfer are used: sender- initiated, receiver- initiated, and sender-receiver-initiated [11].

#### 2.6.2.1 Sender-Initiated Algorithm

In this method, the sender process that wishes to off-load some of its computation initiates the algorithm. The load distribution from heavily load sender to a lightly loaded receiver requires three basic decisions:

- Transfer Policy: If the queue sizes are the only indicator of the workload, a sender can use a transfer policy that initiates the algorithm detecting that its queue length has exceeded a certain threshold upon the arrival of new process.

- Selection Policy: The newly arrived process would be a natural candidate for the selection policy since no preemption is necessary for removing the process. A sender must probe other nodes to decide on a suitable receiver.

- Location Policy: A simple probing strategy is to poll a certain number of nodes one a time and select the node with the smallest as a target receiver [11].

The sender-initiated algorithm works better when the load in the network is not high; in this case, it is easy for the sender to find a receiver.

#### 2.6.2.2 Receive-Initiated Algorithm

A receiver can pull a process from others to its site for execution. The receiver-initiated algorithm can use the same transfer policy method which starts the pull operation when its queue length falls below a certain threshold.

It is possible to combine the two algorithms together, for instance, a node can start the sender-initiated algorithm when its queue exceeds the threshold, on the other hand, it can activates receiver-initiated algorithm when its queue falls below the threshold [11].

# CHAPTER 3

# TOPOLOGY MANAGEMENT

In the light of existing research in the field, the concentration of our work is shifted towards developing new approaches for topology management and task allocation. In this chapter, we cover topology management. We present two new cooperative computing techniques, Dynamic Server Selection (DSS) and Clustering Scheme (CS) for the topology management over MANETs. The main objective of DSS and CS is to classify the nodes into servers and non-servers, based on the energy of the node. In order to make this classification, the node's energy is compared to that of other nodes which stand a distance of at most three-hops from it. This restriction simplifies the selection scheme and reduces its energy consumption and required time, thereby reducing the overall selection cost [9]. Before starting the discussion of the schemes, some definitions need to be introduced to make a solid understanding of the system model.

**Definition 3.1: Node Accessibility**

The relation $i \Rightarrow j$ describes single-hop accessibility between node $i$ and node $j$. We assume that the link between any two nodes is symmetric, therefore, if $i \Rightarrow j$ then $j \Rightarrow i$; represented as j $\Leftrightarrow i$. Multi-hop accessibility between two nodes $i$ and $j$ ($i \overset{p}{\Leftrightarrow} j$), is given iff node $i \in N$ is currently able to send information to node $j$ via $p$ hops connection. Hence if $i \overset{0}{\Leftrightarrow} j$ is given, then i= j.

**Definition 3.2: Neighborhood Relationship**

Node *j* and node *i* are called neighbors iff they both belong to the same network and are capable of sending information to each other via a single hop connection. This relationship is expressed as: $i, j \in N \wedge i \Leftrightarrow j$. To denote the neighbors of node *i*, we say: *Neig(i)={...}*, where the right side is the set of the neighbors of node *i*. If *i* has no neighbors, then *Neig(i)=ϕ or { }*.

**Definition 3.3: Network lifetime**

Without losing generality, the lifetime of the network is the time until the first node dies.

## 3.1 Dynamic Server Selection (DSS)

We will use the simple topology illustrated in Fig. 3.1 to explain the selected algorithm. The line that connects any two nodes indicates that they can hear each other, and the numbers signifies the energy level of each node. For simplification purposes, these numbers are used in our explanation as the names of the nodes. Selection scheme consists of five steps:

1- At first, neighboring nodes exchange information about their energy levels, the fact that, in turn, enables each node to collect information about its neighbors.

2- Each node compares its energy level to that of its neighbors. If it possesses the highest energy level among its neighbors, the node selects itself to be a server node, otherwise, it does nothing. For instance, in Fig. 1, the neighbors of node 50 are 25, 30, and 20. Since node 50 has the highest energy level among its neighbors, it selects itself as a server node. In contrast, node 55 cannot be a server node,

because it has two neighbors with higher energy levels, being 60 and 70.

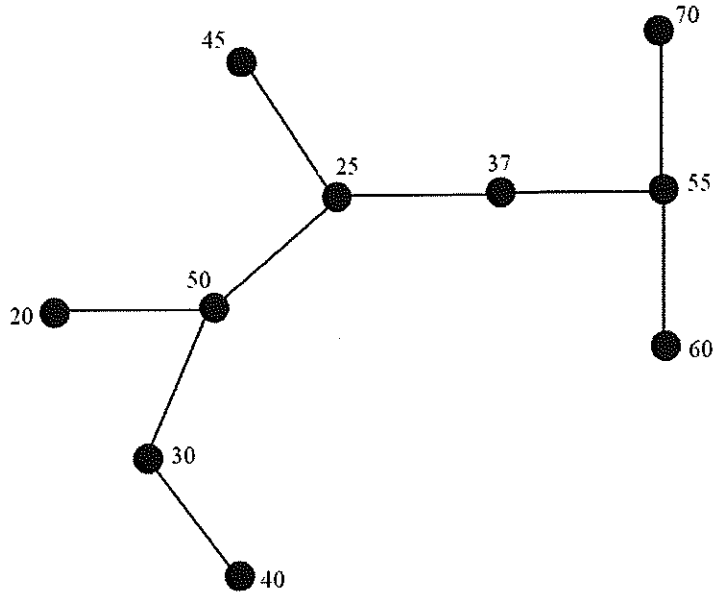



Figure 3.1: Simple MANET Topology

At this stage, i.e. when the server nodes are selected, the network is divided into two types of nodes: server and non-server nodes. In our network, nodes 70, 60, 50, 45, and 40 are server nodes, whereas the others are non-server nodes. As depicted by Fig.3. 2.

**Definition 3.4: Server Node**

Node *(i)* is a server node iff $E_i \geq E_k \ \forall k \in Neig(i)$. We use $S_i$ to indicate that node *(i)* is a server node.

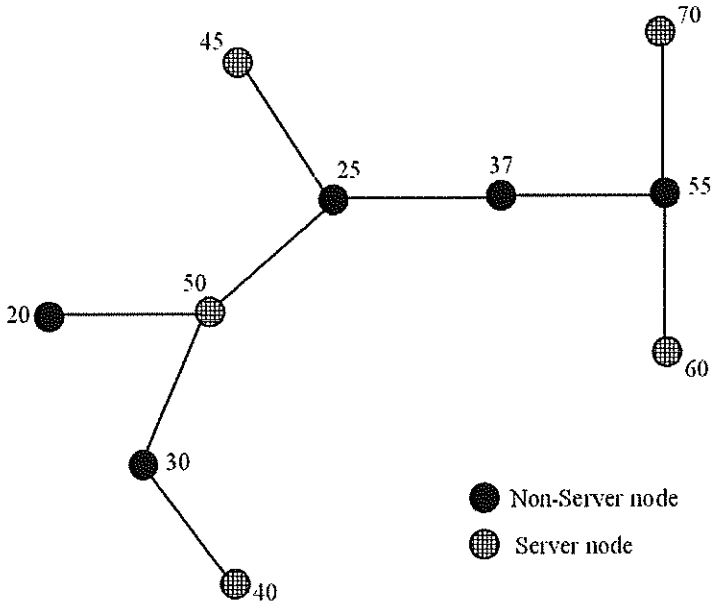



Figure 3.2: Server and non-server nodes

3- At this step, each server node broadcasts a message to its neighbors to announce itself as being a server. For instance, in Fig. 3.2, node 55 will be informed about the servers 60 and 70.

In the following two steps, servers divide themselves into two categories: Servers at level one and servers at level two.

4- When a non-server node receives a message from one (or more) server node(s), it broadcasts the name of the server with the highest energy level to its neighbors. For example, node 55 broadcasts the name of server 70, while node 25 broadcasts the name of server 50.

Notice that after this step, each server has been informed about the servers that are three or less hops far from it. For example, server 60 has been informed about server 70, and vise versa.

5- Each server compares its energy level with that of all the known servers. In the figure above, server 70 compares its energy level with server 60, and since it has a higher energy level, it selects itself as server at level one. On the other hand, server 60 compares its energy level with server 70, and selects itself as server at level two. In the same way, servers 45 and 40 are at level two, whereas server 50 is at level one as showed by Fig. 3.3.

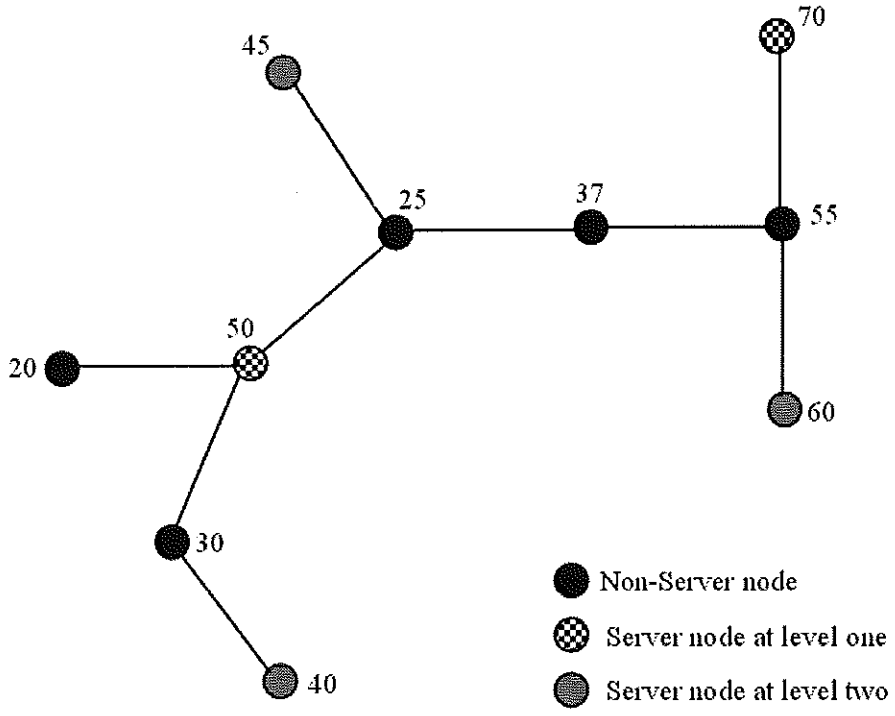



Figure 3.3: Servers at level one and servers at level two

**Definition 3.5: Server at Level One**

Node *(i)* is a server at level one iff $E_i > E_j, \forall j \overset{p\leq 3}{\Rightarrow} i$ we use $S_{i1}$ to denote that node *(i)* is a server at level one.

**Definition 3.6: Server at Level Two**

Node *(i)* is a server at level two iff:

$(E_i > E_j, \forall j \in Neig(i)) \wedge (\exists k \overset{p\leq 3}{\Leftrightarrow} i, E_k > E_i)$ We use $S_{i2}$ to denote that node *(i)* is a server at level two.

The main differentiating point between "servers at level one" and "servers at level two" lies in the remote load they can receive. In this sense, a restriction is put to their remote load. Since servers at level one have higher energy levels than those of servers at level two, their threshold of the remote load is greater, implying that servers at level one will receive more remote load.

## 3.2 Clustering Scheme (CS)

Clustering scheme structures the nodes in the network with a siblinghood and parenthood relationship hierarchy. The simple topology illustrated in Fig. 3.4 will be used to explain the clustering algorithm. Clustering algorithm consists of two main phases:

**Phase 1**

1- Neighbor nodes exchange information about their energy levels. Each node tends to select the neighbor with the highest energy to be its *parent*. To express the *parenthood* relationship between $i$ and $j$, the symbol $i \rightarrow j$ is used, showing that $j$ is a parent of $i$. On the other

hand, the relationship of the node with the other remaining neighboring nodes is called *siblinghood* relation. We use (i —— j) to denote that *i* and j are siblings. Fig. 3.5 shows a topology in which nodes are connected by parenthood relationship (single arrow line) and siblinghood relationship (line).

At this stage, it is important to know what the real functions of a parent and a sibling are. These functions are explained in Definitions 3.7 and 3.8, respectively.

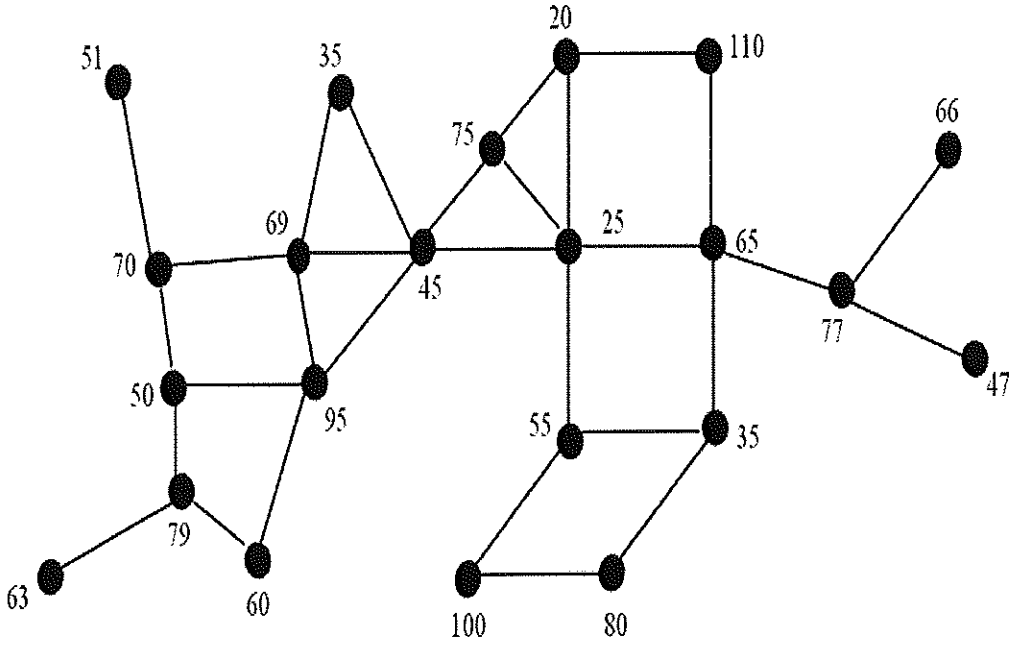



Figure 3.4: Simple MANET Topology

**Definition 3.7: Parent Function**

The parent function can be defined over N nodes as follows:

$$p(i)=\{j \text{ where } j \in Neig(i) \text{ and } (E_j \geq E_k \ \forall \ k \in Neig(i))\} \qquad \textbf{(3.1)}$$

The parent function can also be expressed in terms of a child function (ch). In other words, $p(i)=j$, is equivalent to $ch(j)=i$.

**Definition 3.8: Sibling Function**

The sibling function can be defined over N nodes as follows:

$$s(i)=\{j \text{ where } j \in Neig(i) \text{ and } p(i) \neq j \text{ and } p(j) \neq i\} \qquad \textbf{(3.2)}$$

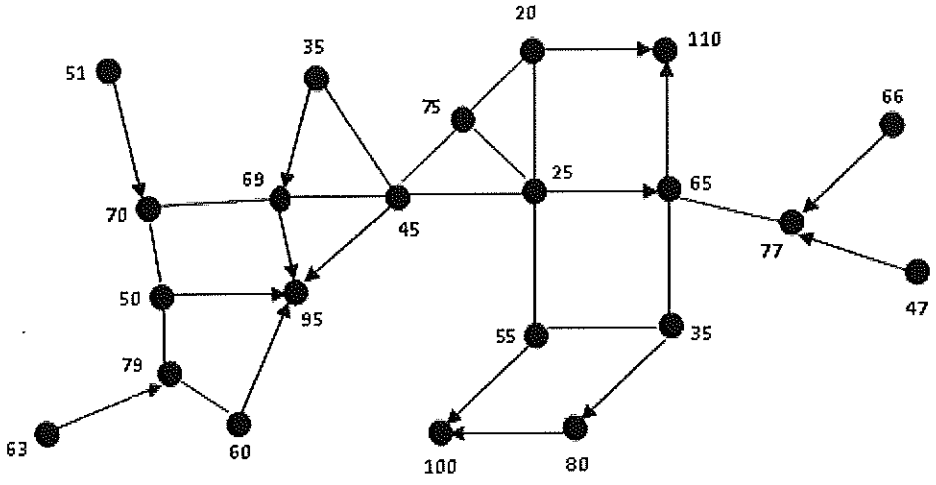



Figure 3.5: Parenthood and Siblinghood relationships between nodes

2- In this step, each node broadcasts its parent's information to its neighbors, the fact that allows each node to collect information about all its neighbor's parents.
3- When the node gets information about all the parents that surround it, it chooses the one with the highest energy level and broadcasts the information about it. After that, each parent builds a vector that contains all the parents that it has known. Fig. 3.6 reveals this fact by a list of parents shown by at parent.

Notice that in Fig. 3.6 the nodes 80 and 65 are parents of nodes 35 and 25, respectively. Yet, because they also play the role of a child for nodes 100 and 110, respectively, they did not build a vector about the parents of their neighbors.

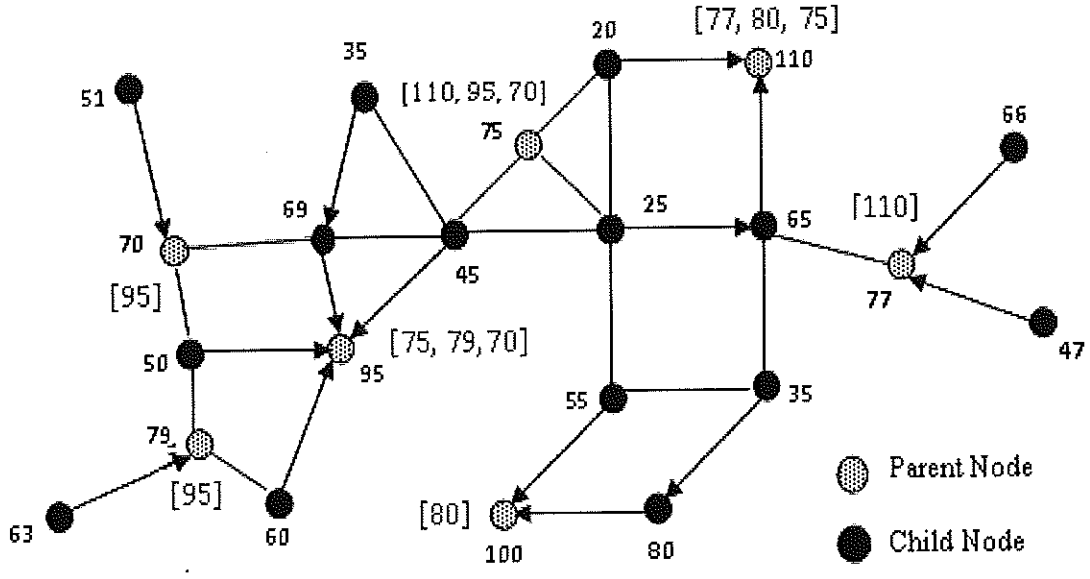



Figure 3.6: Each parent with its collected information

**Phase 2**

In order to choose its upper parent, each parent selects the parent with the highest energy level from its vector. For example, parents 75 and 70 choose parents 110 and 95 to be their upper parents, respectively. This convention is shown in Fig. 3.7.

These parents are then called intern parents, and $P_{in}$ denotes the set of those intern parents, whereas the set of upper parents are denoted by $P_{upp}$. Note that each upper parent is also an intern parent; however, the inverse is not true.

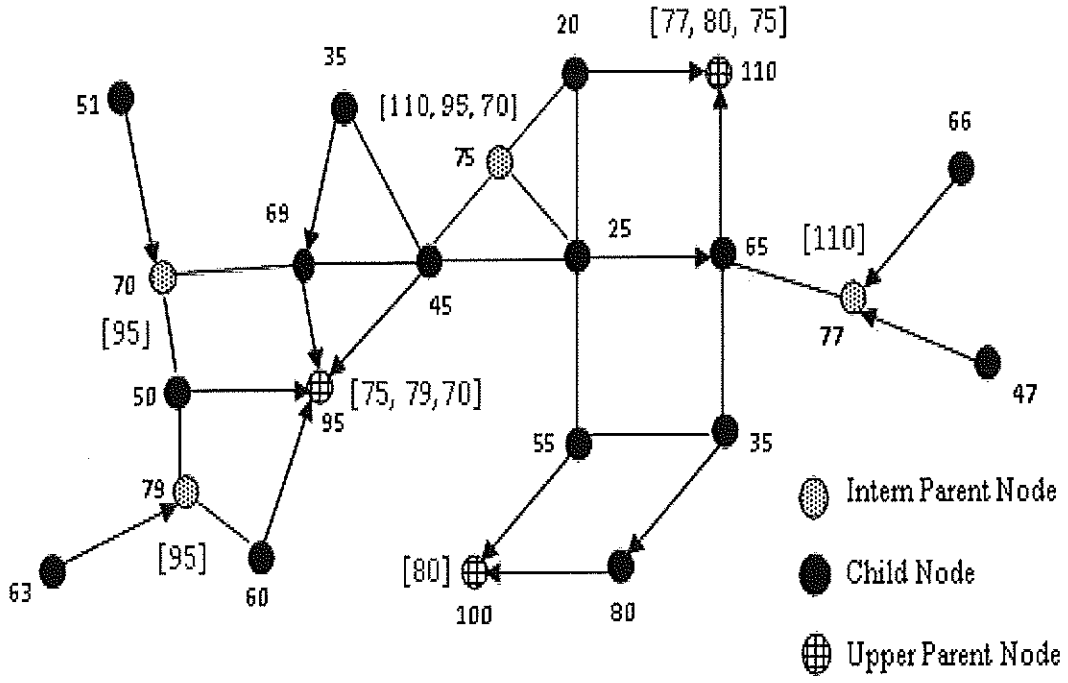



Figure 3.7: Intern Parent and Upper Parent nodes

Hence, upon accomplishing this algorithm, the nodes can be divided into four classes:

- Child nodes: Those that have direct parents
- Parent nodes: Those that have direct children and direct parents
- Intern parents: Those that have upper parents
- Upper parents: Those that do not have any parent.

Fig. 3.8 shows the child nodes, parents, intern parents and upper parents.

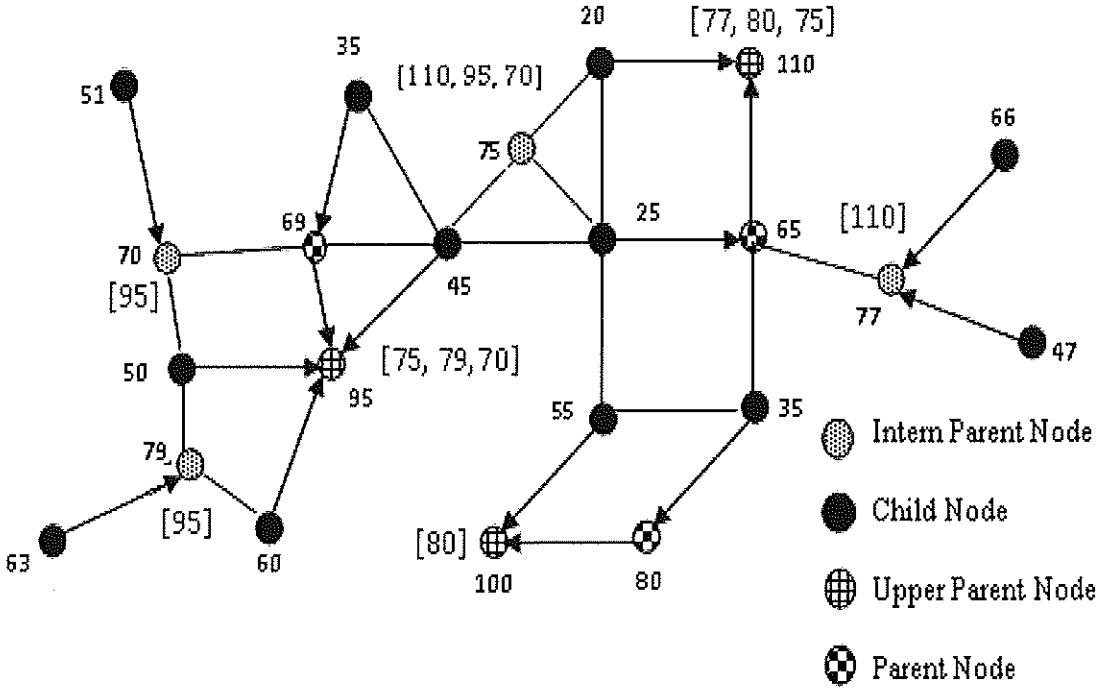



Figure 3.8: Intern Parent, Upper Parent and Parent nodes

**Definition 3.9: Upper Parent Function**

The upper parent function can be defined over N intern parent as follows:

$$p_{upp}(i)=\begin{cases} j:(i,j\in P_{in})\wedge(i\overset{p\leq 3}{\Leftrightarrow}j)\wedge(E_j>E_k,\forall k\in P_{in}\wedge i\overset{p\leq 3}{\Leftrightarrow}k) \\ i:(E_i)\geq E_k,\forall k\in P_{in}\wedge i\overset{p\leq 3}{\Leftrightarrow}k) \end{cases} \quad \textbf{(3.3)}$$

If $p_{upp}\ (i)=i$, then $i$ is called upper parent.

As mentioned in phase 1, an intern parent with its children constitutes a cluster. Therefore, in order to distinguish these clusters from the clusters of upper parents, they are called **intern clusters.**

**Definition 3.10: Intern Cluster**

The intern cluster is denoted by $C_{in}$, meaning that the intern cluster of the intern parent $i$, $C_{in}(i)$, is the group of nodes with their children that share node $i$ as a common parent. Therefore $C_{in}(i)$ is defined as:

$$C_{in}(i) = \{j, k, \ldots n\}, \forall j, k, \ldots n, \quad \textbf{(3.4)}$$

Such that $P(j) = i \wedge P(k) = i \wedge \ldots\ldots \wedge P(n) = i$,

For example, in *Fig. 3.8*:

$C_{in}(77)$= {77,66,47}, $C_{in}(79)$={79, 63}, and $C_{in}(75)$={75}.

At this point, it can be said that each upper parent with its children and intern parents constitute a cluster. Fig. 3.9 shows each upper parent with its cluster.

**Definition 11: Upper Parent Cluster**

The cluster of upper parent $i$, $C_{upp}(i)$, is the set of all intern clusters whose upper parent is $i$:

$$C_{upp}(i) = C_{in}(j) \cup C_{in}(k) \cup \ldots\ldots \cup C_{in}(n), \forall j, k, \ldots n \quad \textbf{(3.5)}$$

Such that $p_{upp}(j) = i \wedge p_{upp}(k) = i \wedge \ldots. \wedge p_{upp}(n) = i$ .

For example, in Fig.3.9, $C_{upp}(95) = C_{in}(95) \cup C_{in}(79) \cup C_{in}(70)$

**Definition 3.12: Set of all Clusters**

The set C of all clusters is defined as follows:

$$C = C_{upp}(i) \cup C_{upp}(j) \cup .... \cup C_{upp}(n) \qquad \textbf{(3.6)}$$

Where $i$ , $j$ ,... $n \in P_{upp}$

For example, in *Fig. 3.9*, $C = C_{upp}(110) \cup C_{upp}(100) \cup C_{upp}(95)$

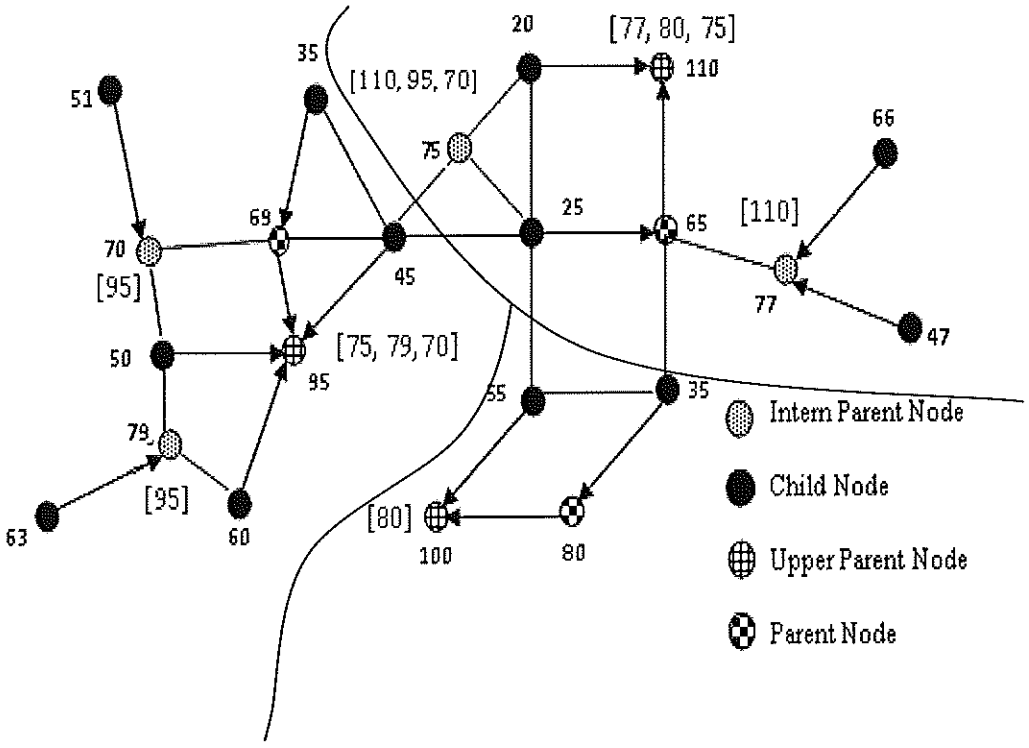



Figure 3.9: Upper parent's clusters

# CHAPTER 4

# ENERGY AWARE DYNAMIC TASK ALLOCATION

In this chapter, we present two energy aware dynamic task allocations called the Energy Aware Dynamic Servers Selection and Task Allocation (EADSSTA) and Scalable Energy-aware Dynamic Task Allocation SEADTA. The objective of these two algorithms is to minimize both tasks execution time and energy by using the proposed topology management techniques DSS and CS.

## 4.1 System Model

The system in the MANET topology consists of $N$ nodes which differ in their processing capabilities and can forward messages to their neighboring nodes. Each node has the following entities:

- Processing identification: identifies the processor type,
- Waiting queue for tasks to be executed,
- $E_i$: Energy level, where i is the node id.
- T[i,j]: execution time for all types of tasks of all the nodes, and
- E[i,j]: execution energy for all types of tasks of all the nodes, where $i$ is the task type and $j$ is the processor type

It is assume that there are $n$ types of tasks generated in each node and each task comes with:

- id number, which identifies the task type,
- Data to be processed, amount of which is selected randomly between the minimum and the maximum data size.

## 4.2 The EADSSTA Task Allocation

The problem that is solved by EADSSTA can be described as follows:

The network lifetime is divided into many rounds. At the beginning of each round, the DSS is used to locate the best new remote servers which will maximize the minimum residual energy at the end of the round. Nodes that want to execute a task remotely will follow the scenario explained in the remote task section 4.2.1 in order to select one of the available servers.

### 4.2.1 The EADSSTA Remote Task Allocation

When a task arrives at a node, it enters into the waiting queue. However, some tasks may not be processed due to processing incapability, energy shortage or load balance. In such cases, the node must send inquiries to the other nodes in the network. This follows a remote allocation algorithm that contains the following steps:

- The sender node broadcasts a request message in the network and initiates a request timer. This request is called B-REQ(SN,TSK_TYP, Hops), where SN is the sequence number of the sender node, TSK_TYP is the task type, and Hops field is used to restrict flooding of this message (see Algorithm in Fig 4.1).
- Only the servers (i.e. servers at level one, and those at level two which have processing capability to execute the task and possess a threshold load greater than the remote loads) reply to the broadcasted request message by sending RLP(mySN,yourSN,MY_INFO), where mySN is the server's sequence, yourSN is the destination sequence number, and MY_INFO contains the server's information (see Section 4.5). The role of the other nodes is to forward the broadcasted requests and replies (see algorithm Fig 4.2).

After the timer expiration, the sender node, based on the objective function in Section 4.5, computes the best server that can execute the task, then sends the

task to the best server by sending a message TSK(SN,TSK_INF), where SN is sequence number and TSK_INF is the task information.

When a task arrives to a node

The node checks its own recourses for:

- processing time of the task
- battery level
- queuing time

if the task can be processed

put the task in my waiting queue

else

Broadcast B-REQ(SN,TSK_TYP, Hops) inquiry message

Start inquiry timer()

Receive RLP(mySN,yourSN,MY_INFO) replies

After timeout

Evaluate the best candidate according to the objective function

Send the task TSK(SN,TSK_INF) to the best candidate

Figure 4.1: Task allocation algorithm at the sender node (EADSSTA)

When a node receives a message

If the message is a new B-REQ(SN,TSK_TYP, Hops) and Hops is greater than zero

If the node is a server

Decrease Hops by one

Forward B-REQ(SN,TSK_TYP, Hops) message

Record the current request in the request list

Go to (1)

Else

Decrease Hops by one

Forward B-REQ(SN,TSK_TYP, Hops) message

Else if the message is TSK(SN,TSK_INF) destination to the node

Receive the message and execute the task

Else if the message is TSK(SN,TSK_INF) destination to

another node

Network layer takes the suitable action

(1)

The node checks the recourses:

- Processing capability
- Load

If node can execute the task

Sends RLP(mySN,yourSN,MY_INFO)

Else

Do nothing

Figure 4.2: Task allocation algorithm at the intermediate/receiving node (EADSSTA)

### 4.3 The SEADTA Task Allocation

The problem that is solved by SEADTA can be described as follows:

The network lifetime is divided into many rounds. At the beginning of each round, use CS to locate the best new parents (remote servers) which will maximize the minimum residual energy at the end of the round. Nodes that want to execute a task remotely will follow the scenario explained in the remote task section 4.3.1 in order to select one of the available parents.

#### 4.3.1 The SEADTA Remote Task Allocation

When a task arrives at a node, it enters into the waiting queue. However, some tasks may not be processed due to processing incapability, energy shortage or load balance. In such cases, the node must send inquiries to the other nodes in the network.

The remote allocation algorithm contains the following steps:

- If the sender node has a parent (a direct parent, a intern parent, or an upper-parent), it sends a request message to its parent and initiates a request timer (see the algorithms in Fig 4.3 and Fig 4.4). This request is called as REQ(SN,TSK_TYP), where SN is the sender node's sequence number, and TSK_TYP is the task type. After the timer expiration, if the parent sends a reply message, then the node sends the task to it via a message. The message sent by the parent is represented as RLP(mySN,yourSN,MY_INFO), where mySN is the sender's parent sequence, yourSN is the destination sequence number, and MY_INFO contains the sender's information (see Section 4.5), whereas the message sent by the node is TSK(SN,TSK_INF), where SN is sequence number and TSK_INF is the task information. If the parent does not send any reply, then the sender node broadcasts a request in the network B-REQ(SN,TK_TY,Hops), where the Hops field mentioned in the B-REQ message is used to restrict flooding of this message.

- If the sender node is an upper-parent, it broadcasts a request B-REQ(SN,TSK_TYP, Hops) in the network and initiates a request timer.

- Only the upper parents and the intern parents that have the processing capability to execute the task, reply to the broadcast request by sending RLP(mySN,yourSN,MY_INFO). The role of the other nodes is to forward the broadcasted requests and replies (see the algorithm in Fig 4.5). After the timer expiration, the sender node computes the best node that can execute the task and sends TSK(SN,TSK_INF) to it .

**Theorem 4.1**

SAEDTA outperforms EADSSTA in terms of energy communication if at least one parent accepts to schedule a task from one of its children.

**Proof.**

Assume that the number of parents (in all levels) in SAEDTA is equal to the number of servers (in level one and two) in EADSSTA. Let this number be S, where S>1.

Assume that there are $n$ nodes which want to execute a task remotely, and n>1. Let only one parent, in SAEDTA accept to execute its child's task. Now (n-1) will broadcast the messages in the network, and the other one will schedule its task in its parent. Therefore, the cost will be 1+(n-1)*S.

In the case of EADSSTA, however, all $n$ nodes will broadcast in the network. As a result, the cost will be n*S, which is obviously greater than 1+(n-1)*S, i.e.
$1 + (n-1) * S < n * S$

When a task arrives to a node

The node checks its own recourses for:

- processing time of the task
- battery level
- queuing time

if the task can be processed

put the task in my waiting queue

else

if the node has a parent

checks if the parent is here

send request REQ(SN,TSK_TYP) to the parent

go to (2)

else

go to (1)

else

if the node is upper parent

go to (1)

Figure 4.3: Task allocation algorithm (part 1) at the sender node (SEADTA)

```
(1)
   Broadcast B-REQ(SN,TSK_TYP, Hops) inquiry message
   Start inquiry timer()
    Receive RLP(mySN,yourSN,MY_INFO)  replies
   After timeout
     Evaluate the best candidate according to the objective function
      Send the task TSK(SN,TSK_INF) to the best candidate


(2)
  start inquiry timer()
   receive RLP(mySN,yourSN,MY_INFO)  reply from the parent
   after timeout
     if the parent has replied
         send the task TSK(SN,TSK_INF)  to the parent
     else
   go to (1)
```

Figure 4.4: Task allocation algorithm (part 2) at the sender node (SEADTA)

When a node receives a message

If the message is a new B-REQ(SN,TSK_TYP, Hops) and Hops is greater than zero

If the node is an upper parent or intern parent

Decrease Hops by one

Forward B-REQ(SN,TSK_TYP, Hops) message

Record the current request in the request list

Go to (1)

Else

Decrease Hops by one

Forward B-REQ(SN,TSK_TYP, Hops) message

Else if the message is REQ(SN,TSK_TYP) from the child

Go to (1)

Else if the message is REQ(SN,TSK_TYP) destination to another node

Network layer takes the suitable action

Else if the message is TSK(SN,TSK_INF) destination to the node

Receive the message and execute the task

Else if the message is TSK(SN,TSK_INF) destination to another node

Network layer takes the suitable action

(1)

The node checks the recourses:

- Processing capability

If node can execute the task

Sends RLP(mySN,yourSN,MY_INFO)

Else Do nothing

Figure 4.5: Task allocation algorithm at the intermediate/receiving node (SEADTA)

## 4.4 Task Delay and Energy Consumption

There are two factors that contribute to the overall task delay, the queue delay and the communication delay. The queue delay is the time interval from putting the task in the waiting queue until the time it leaves it in order to be processed. The communication delay consists of the request-reply time and the time needed for task to move from the source node to the destination node. Therefore, Task delay can be calculated as follows:

Task delay = queuing delay + request-reply delay+

task's transfer time . **(4.1)**

The energy consumption includes the energy consumed for the task's execution, the request-reply, the task's transfer from the source to the destination node, and the overhearing nodes that are within the range of transmitting nodes along the communication path from sender to receiver. We use the NS2 [10] energy model.

Energy Consumption = task execution energy + request-reply energy +

task transfer energy + overhearing energy **(4.2)**

## 4.5 Objective Functions

When a server (at level one or two) in EADSSTA, or an upper parent (or an intern parent) in SEADTA receives a broadcast request, it records the request in the request list and then sends its reply. The request list is a table that contains the following fields about requests:

- The requester-node address
- The distance to the requester (in hops)
- The expiration time.

The requested node learns the requester-node address and the distance to it from the request message (common header and ip header message in NS2 simulator). As for the expiration time, it is estimated on the basis of the distance that separates the requested node from the requester node. The request is deleted from the request list either when the expiration time is passed, or when the node receives a remote task from the requester node.

Every upper parent, intern parent, server at level one, and server at level two replies to the broadcast request by sending the message represented as RLP(mySN,yourSN,MY_INFO). The MY_INFO field in the message contains the following information about the requested node: energy level, processing type, waiting queue length and current registered requests from the nodes that have shorter distance to the requested node than the current requester node.

When the requester node receives replies from the requested nodes, it will compute the best candidate to be the receiver of the task based on the following objective function:

Assume that the requester node is node ($i$), the requested node is ($k$), and the task to be remotely allocated is ($j$).

$$\text{Cost}=\min[\, T_{j,k} + QL_k + E_k + COM_{i,k} + ECOM_{i,k} + \sum_{\forall n, n \overset{p}{\Leftrightarrow} k <= i \overset{p}{\Leftrightarrow} k} REQ_{n,k} \,] \qquad \textbf{(4.3)}$$

for all node k that replied to node ($i$) where:

$T_{j,k}$ is the execution time of task ($j$) on node ($k$),

$QL_k$ is the waiting queue length on node k,

$E_k$ is the energy level on node k,

$COM_{i,k}$ is the communication time needed to send task ($j$) from node ($i$) to node ($k$)

$ECOM_{i,k}$ is the communication energy needed to send task ($j$) from node ($i$) to node ($k$)

$\sum_{\forall n, n \overset{p}{\Leftrightarrow} k <= i \overset{p}{\Leftrightarrow} k} REQ_{n,k}$ shows the registered requests in node ($k$) from all nodes ($n$) that have shorter distance to node ($k$) than node ($i$).

Since the energy and the time have different units, their elements in the objective function need to be normalized:

$$T_{j,k} = T_{j,k}/T_{max} \qquad \textbf{(4.4)}$$

where $T_{\max}$ is the maximum processing time cost for task ($j$) among all the replied nodes.

$$QL_k = \begin{cases} \infty & / Q_k > Th \\ Q_k / Th & / otherwise \end{cases}, \qquad \textbf{(4.5)}$$

where $Q_k$ is the waiting queue length on node ($k$), and $Th$ is the queue threshold on node ($k$).

$$E_k = \begin{cases} \infty & /(E_k - E_{j,k}) <= Th \\ E_{j,k} / E_{\max} & / otherwise \end{cases}, \qquad \textbf{(4.6)}$$

Where $E_k$ is the energy level on node ($k$), $E_{j,k}$ is the energy needed to execute task ($j$) on node ($k$), and $E_{max}$ is the maximum energy cost needed to execute task ($j$) among all the replied nodes.

$$COM_{i,k} = COM_{i,k} / COM_{\max} \qquad \textbf{(4.7)}$$

Where *is* $COM_{max}$ is the communication time to send task ($j$) from node ($i$) to the farthest node among the replied nodes.

$$ECOM_{i,k} = ECOM_{i,k} \Big/ ECOM_{\max} \qquad \textbf{(4.8)}$$

Where $ECOM_{max}$ is the energy to be consumed to send task ($j$) from node ($i$) to the farthest node among the replied nodes.

$$= \sum_{\forall n, n \overset{p}{\Leftrightarrow} k \le i \overset{p}{\Leftrightarrow} k} REQ_{n,k} \Big/ MAX\left( \sum_{\forall n, n \overset{p}{\Leftrightarrow} m \le i \overset{p}{\Leftrightarrow} m} REQ_{n,m} \right) \qquad \textbf{(4.9)}$$

Assume that node ($m$) is one of the replied nodes that have received the maximum requests from all nodes that have shorter distance to it than node ($i$). Therefore, $MAX\left( \sum_{\forall n, n \overset{p}{\Leftrightarrow} m \le i \overset{p}{\Leftrightarrow} m} REQ_{n,m} \right)$ is the number of registered requests in node ($m$) from all nodes ($n$) that have shorter distance to it than node ($i$).

# CHAPTER 5

# ENERGY AWARE STATIC SCHEDULING

## 5.1 Introduction

In literature, the problem of static task scheduling, in its general form, has been proved to be NP-complete [11, 12, 13], and many heuristic static task scheduling algorithms have been proposed by the researchers. List scheduling algorithms (LS) is one of the most important classes of the heuristic algorithms, which is used in both homogenous systems with identical processors, and heterogeneous systems [11, 12].

Recently, LS was used in the energy aware static scheduling by some researchers, such as [14, 15]. Authors in [14] used LS algorithm to study two scheduling problems, the minimizing scheduling length with energy consumption constrain and the minimizing energy consumption with scheduling length on DVS multiprocessor computers. On the other hand, study in [15] proposed an energy aware static scheduling (ETS) using LS algorithms, that aims to minimize both the execution time and the energy. ETS have not studied the network life time and the topology formation. The study in [16] has developed a two duplication energy aware static task scheduling for homogenous mobile clustering, but the cluster formation was absent in this study.

Most of the previous studies have disregarded the overall network lifetime, the heterogeneity, and the frequent changes of the topology in the MANET. In this section, two new energy aware static task scheduling models are presented they aim to maximize the network lifetime and tackle the problem of heterogeneity and the dynamic change of topology. With the best of our knowledge, this is the first work that studies these issues from a task scheduling

prospective. Therefore, our study comes to participate to fill the above gaps found in literature.

### 5.2 System Model

In static scheduling, parallel applications are represented by a Directed Acyclic Graph (DAG), where vertices indicate the tasks and edges represent the communication and precedence among tasks. It is assume that the parallel applications are generated in each node by the compiler. TGFF [17] tools are used to generate such applications (DAGs).

### 5.3 Task Model and Processor Model

In the proposed model, there are *m* types of tasks that can be generated in each node. Each task is generated along with the following entities:

- Task id which identifies the task type.
- Task size which is selected randomly between the minimum and maximum size.
- Data to be processed which is selected randomly between the minimum and maximum size.
- Data to be send to each of its successor tasks is selected randomly between the minimum and maximum size.

None of the tasks can start its execution until all its precedence tasks have finished their execution and the data sent by these tasks have been received by the node at which the task is to be executed.

The topology environment consists of K type of processors. Processor types are different in their processing capabilities in terms of execution time and energy to be spent in executing each task. Each node has the following entities:

- A unique id which identifies the node.
- Processing identification which identifies the processor type.
- $T_{exe}$ [m,k] matrix which contains the execution time for all task types on all types of processors.
- $E_{exe}$ [m,k] matrix which contains the execution energy for all task types on all types of processors.

We assume that the required time and energy to send one unit of data from any node to any of its neighbors are equal.

Having finished the system modeling, now we are in the place to explain the algorithms.

### 5.4 Energy Aware Static Task Scheduling Using DSS (EA-STS-DSS)

After the servers have been selected in each round, we need to add one more step to the server selection scheme in Section 3.1:

- Each server announces itself by broadcasting a message in the network.
- Each server builds a list contains all known servers.

The objective of the above two steps is to enable each server to compute its routing tables to all known servers. This information is needed in task scheduling algorithm.

#### 5.4.1 The EA-STS-DSS Problem Definition

The network lifetime is divided into equal rounds. At the beginning of each round, new servers are selected to be used in the execution tasks during this round. Parallel applications are generated randomly in some nodes that will

follow the EA-STS-DSS task scheduling scheme in section 5.4.2 to schedule the tasks.

### 5.4.2 The EA-STS-DSS Task Scheduling

When the task graph is generated on one node, the node will execute the following steps to schedule the task graph:

1- It sends a broadcast request (RQ) within the network to get information about the current servers in the topology.

2- It starts a request timer.

3- Only the servers at level one and two will reply by sending their node ids, their processing ids, their current energy levels, and their routing tables to all known servers.

4- After timer expiration, the node will execute the energy aware list scheduling in Section 5.6.

## 5.5 Energy Aware Static Scheduling Using Clustering Scheme (EA-STS-CS)

After the network has been divided into clusters by using the CS, We need to add three steps to CS to enable the upper parent of each cluster to collect information about all parents in its cluster.

- Each three-hope parent broadcasts its upper parent id to its children, then, all the parents (at all levels) belonging to the cluster will have known the upper parent of their cluster.

- Each parent broadcasts its id along with the upper parent id, in order to inform all the parents in the cluster about itself.

- Each parent sends its routing table to all known parents in its cluster to the cluster upper parent.

The goal of the above three steps is to let the upper parent of each cluster collect information about all the existing parents in its cluster.

### 5.5.1 The EA-STS-CS Problem Definition

The network lifetime is divided into equal rounds, at the beginning of each round; new clusters are formed. Parallel applications are generated randomly in some nodes. These nodes will follow the EA-SS-CS task scheduling scheme in section 5.4.2 to schedule the tasks.

### 5.5.2 The EA-SS-CS Task Scheduling

When the task graph is generated on one node, this node will execute the following steps in order to schedule the task:

- If the node knows the upper parent of the cluster, it will send the task graph directly to it, otherwise, it will send the task graph to its parent which will in turn send it to the cluster upper parent.
- Since the upper parent has already collected information about all parents in its cluster, it only executes the energy aware list algorithm in section 5.6.

## 5.6 Energy Aware List Task Scheduling (EALTS)

The list algorithm in this section is a type of the well known list scheduling; however, it uses both time and energy to assign a task to a node.

1- Determine the tasks levels which will be used as a priority level for each task according to the technique explained in section 4.7.

2- At the beginning, the priority queue is initialized to include those tasks that do not have any immediate precedence task. Tasks in the priority queue are sorted based on their level. The task with the highest level is the first to be taken from the priority queue. After scheduling a task, the priority queue is updated by inserting those tasks which their immediate precedence tasks have been scheduled.

3- As long as the priority queue is not empty, do the following:

- Get the highest level task from the priority queue, and call this task $t_h$.
- For each server do the following:
    - Compute the finishing time of $t_h$.
    - Compute the required energy to execute $t_h$ and to receive the data sent to $t_h$ from its immediate precedence tasks.
    - Compute the consumed energy at all nodes in the path in delivering a data from all immediate precedence tasks, and determine the resulting residual energies in all these nodes.
    - Find the minimum residual energy node (RE) among all nodes (server and the node in the path in delivering data).

4- Assign $t_h$ to a server that minimizes the finishing time and maximizes the reaming energy according to objective function in section 5.7.

**Theorem 5.1**

The time complexity of EALTS is $n*\log(n)+S*n^2$.

**Proof.**

Assume that there are *n* tasks to be scheduled and S current available servers. In step 1, when a task is added to the priority queue we need to resort the

tasks in it. Since there are *n* tasks, the cost of this step is n*log(n). In step 2, we compute the execution time and the energy for each task in all servers. For each task we compute the residual energy and the receiving and sending time for all the nodes in the path from all immediate precedence tasks. Therefore, the cost of this step is equal to $S*n^2$. Hence, the overall time complexity of the above algorithm is $n*\log(n) + S*n^2$ .

### 5.7 Cost Function

EALTS uses the following cost function to select the best server (parent) to execute the current task. The cost function is a combination of time and energy.

$$Assign(t_h,s)=\min[\alpha*FH_{tim}-\beta*RE]. \qquad \textbf{(5.1)}$$

Where $FH_{tim}$ is the finishing time of $t_h$ on server s, RE is the minimum residual energy when $t_h$ is executed on server s, $\alpha$ and $\beta$ are constant factors that can be configured to reflect the weight of time and energy in the cost function.

Since time and energy have different units, their elements in the objective function in the above algorithm need to be normalized. Assuming that $FH_{max}$ is the maximum finishing time among all servers and $RE_{max}$ is the maximum residual energy among all nodes, the objective can be normalized as follows:

$$Assign(t_h,s)=\min[(FH_{tim}/FH_{max})-(RE/RE_{max})] \qquad \textbf{(5.2)}$$

### 5.8 Task Level

The above scheduling algorithm is a type of the level priority scheduling, which have been proved in literature to be the closest to the optimal schedule [21]. Therefore, this technique, which is used in the computing task level, has an important effect on the efficiency of the scheduler. Since we do not know to

which processor each task will be assigned, in this study we use an estimation task level which is based on the task execution time.

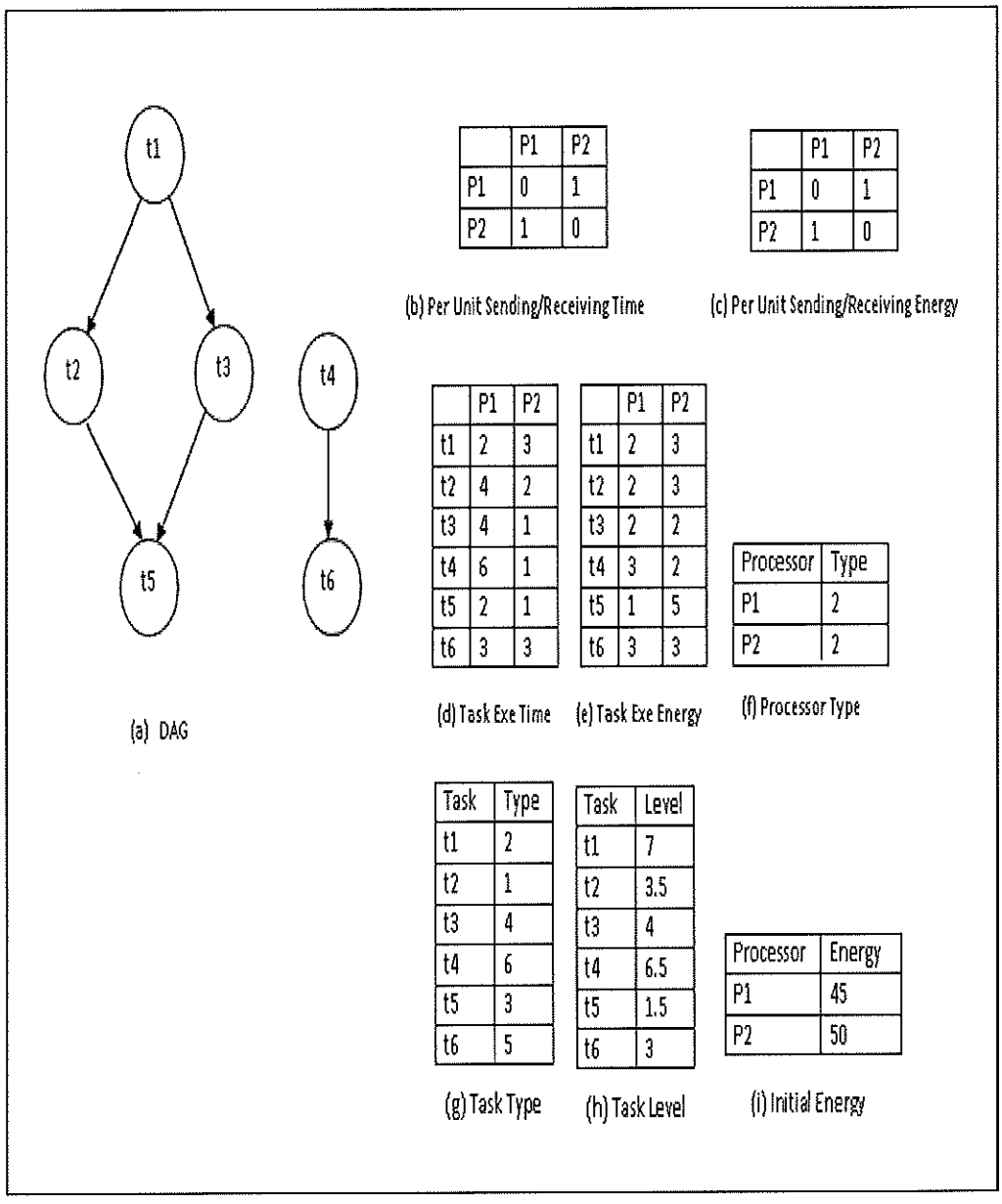
Firstly, we compute the average execution time AVA(t) for all task types on all processor types. Secondly, the level of each task is computed as follows:

$$L(t_i)=\begin{cases} AVA(t_i), \forall\ 1 \leq j \leq n : (t_i, t_j) \notin E \\ AVA(t_i) + max_{e_{ij} \in E}(L(t_j)), \ otherwise \end{cases} \quad (53)$$

If the task is an exit task (task that does not have any successor task), its level is equal to the average execution time of its type. Otherwise, the level of the task is the summation of its type average execution time and the largest task level among its successor tasks.

### 5.9 EA-STS-DSS and EA-STS-CS Behavior

One of the main goals of EASS-DSS and EASS-CS is to prolong the network lifetime. Therefore, we expect the behavior of these tow algorithm to change according to the energy level on the available servers or (parents). To reveal this characteristic we study the model in figure 10. As we can see the initial energy of processors 1 and 2 are 45 and 50 Joule, respectively. The algorithms finish the tasks scheduling (it is set that $\alpha = \beta = 1$) with a makespan equals to 8, and the reaming energy in processor 1 and 2 being 37 and 43, respectively. The scheduling steps are illustrated in Fig 5.1.




| | P1 | P2 |
|---|---|---|
| P1 | 0 | 1 |
| P2 | 1 | 0 |

| | P1 | P2 |
|---|---|---|
| P1 | 0 | 1 |
| P2 | 1 | 0 |

| | P1 | P2 |
|---|---|---|
| t1 | 2 | 3 |
| t2 | 4 | 2 |
| t3 | 4 | 1 |
| t4 | 6 | 1 |
| t5 | 2 | 1 |
| t6 | 3 | 3 |

| | P1 | P2 |
|---|---|---|
| t1 | 2 | 3 |
| t2 | 2 | 3 |
| t3 | 2 | 2 |
| t4 | 3 | 2 |
| t5 | 1 | 5 |
| t6 | 3 | 3 |

| Processor | Type |
|---|---|
| P1 | 2 |
| P2 | 2 |

| Task | Type |
|---|---|
| t1 | 2 |
| t2 | 1 |
| t3 | 4 |
| t4 | 6 |
| t5 | 3 |
| t6 | 5 |

| Task | Level |
|---|---|
| t1 | 7 |
| t2 | 3.5 |
| t3 | 4 |
| t4 | 6.5 |
| t5 | 1.5 |
| t6 | 3 |

| Processor | Energy |
|---|---|
| P1 | 45 |
| P2 | 50 |

Fig 5.1: Task Model, Processor Model, Task levels, and processor Initial energy. (Adapted from [15])

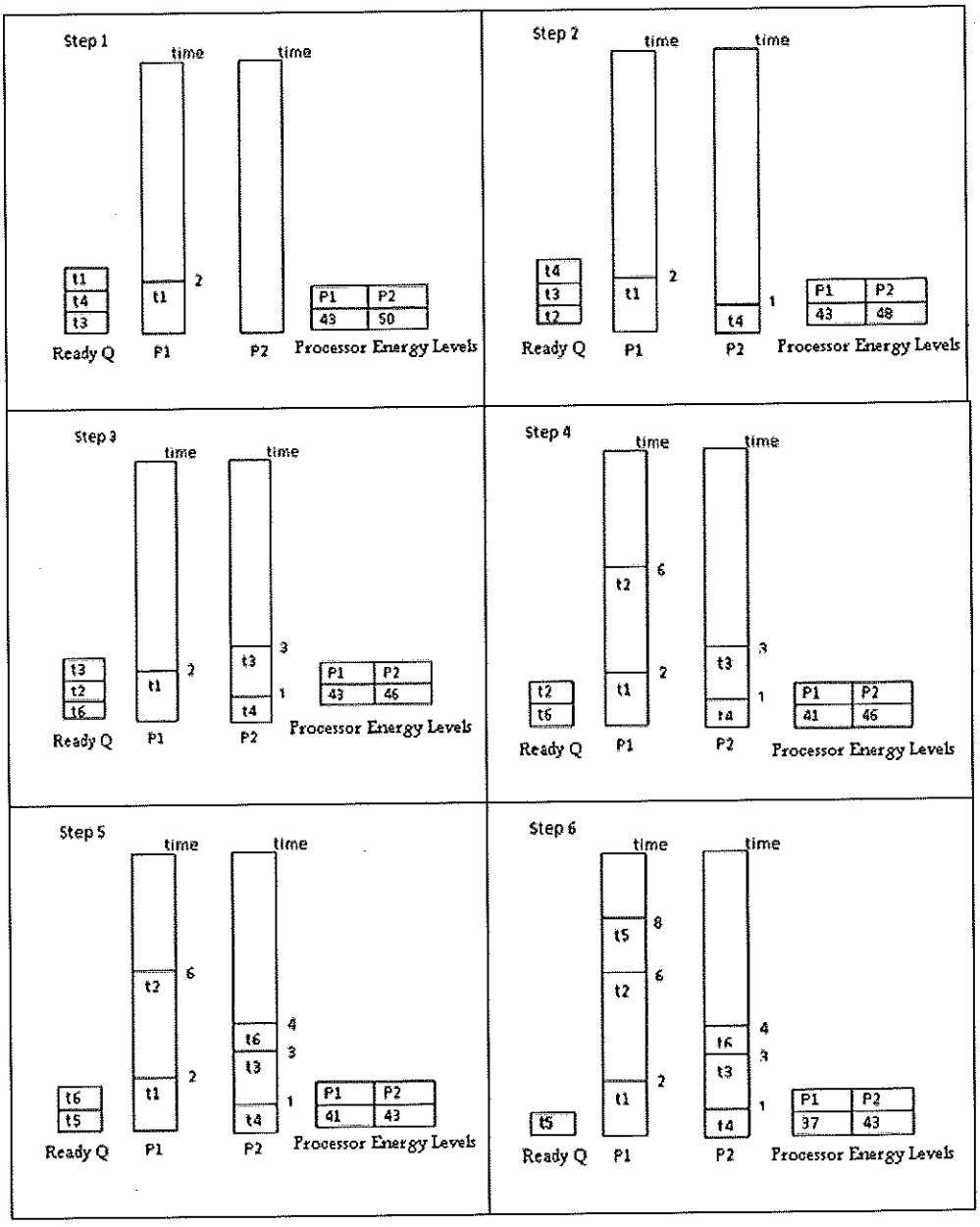



Figure 5.2: Scheduling steps by EA-STS-DSS and EA-STS-CS of the model in fig 5.1 (follows the style in [15])

Assume that the initial energy of processor 1 and 2 is changed to 10 and 40 Joule, respectively. The behavior of the algorithms will change to make a balance in the energy consumption in all the nodes in the network; this fact is shown in Fig 5.3. As we can see, unlike processor 2 which has a high level energy, processor 1 suffers from energy shortage. Therefore, the algorithms adjust themselves accordingly to reflect this fact and they schedule all tasks on processor 2.

Although the makespan is higher than the first scheduling in fig 5.2, the network lifetime is still reaming long. If the time is more important than energy, it is possible to get a shorter makespan by manipulating the value of $\alpha$ and $\beta$ in the cost function.

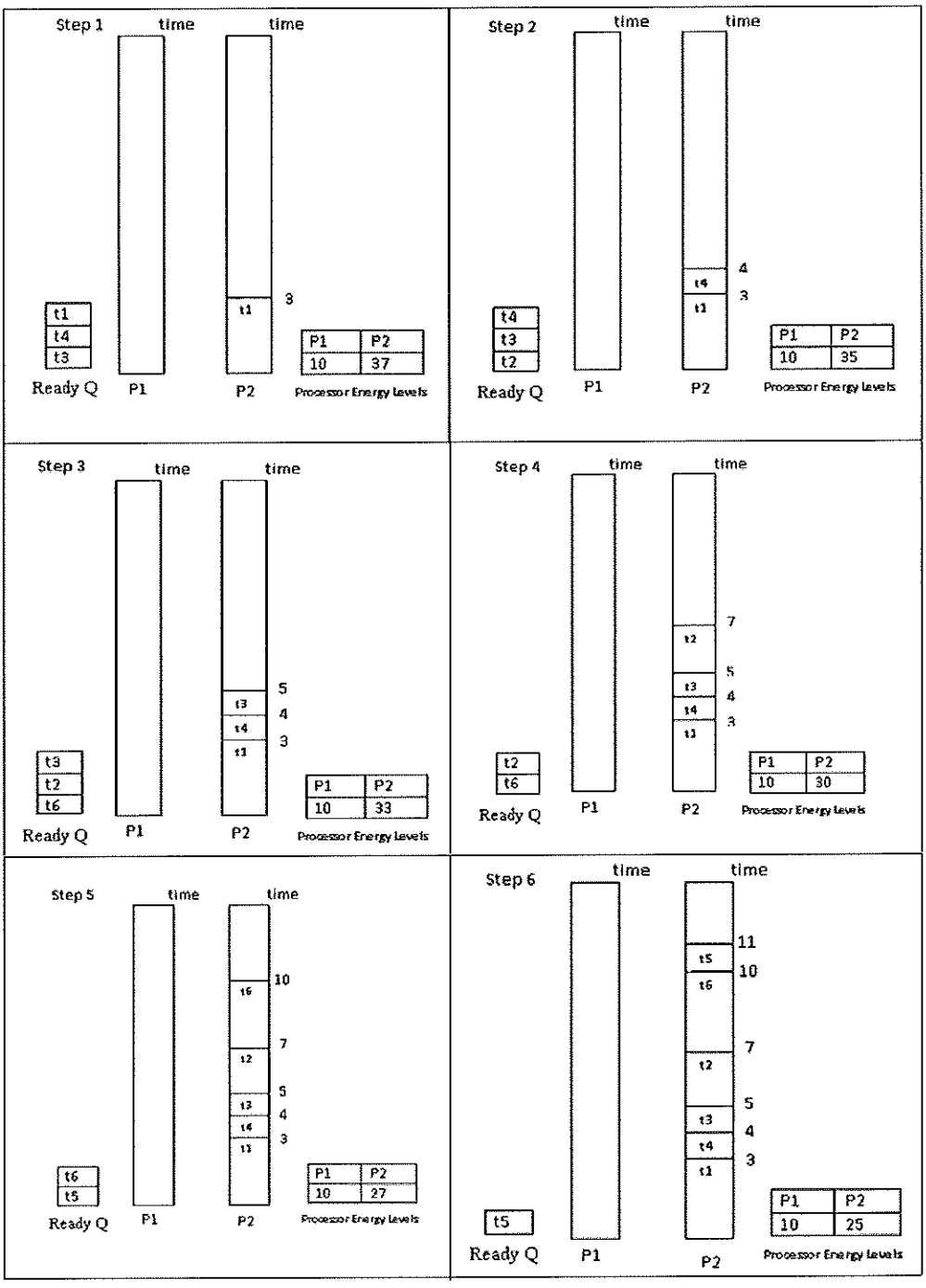



Figure 5.3: Scheduling steps by EA-STS-DSS and EA-STS-CS of the model in fig 13 after initial energy changed

# CHAPTER 6

# PERFORMANCE EVALUATION

To evaluate our models, we extended NS2 simulator to conduct several experiments. In case of the dynamic energy aware task allocation we compare the results with those of the EADTA model in [18], in which the assumptions of task and processor models are similar to those in our dynamic models. In EADTA, inquiring all the nodes in the network by the sender node is an essential requirement for remote allocation.

We compare the result of the static energy aware with those of the ETS model in [15]. ETS did not study the topology management and the method by which any node in the network can collect the information about the system. Moreover, ETS objective was to minimize the makspan time and energy consumption, however, the aims of our static model are prolonging of the network lifetime and minimizing the makspan time.

## 6.1 Simulation Assumptions

The main assumptions of the simulation are:

- Nodes are in a cooperative mode such that each node is willing to propagate messages related to proposed algorithms and execute remote tasks.
- All nodes have enough energy to remain alive until the simulation finish. This assumption ensures that all nodes participate in each experiment.
- The communication between each pair of nodes is symmetric.

- In all experiments, Nodes use the Destination-Sequenced Distance Vector (DSDV) routing protocol to exchange messages. DSDV is a proactive hop-by-hop distance vector routing protocol, where each node broadcasts its routing updated information periodically [1].

- The IEEE 802.11 MAC is used in all experiments and the transmission range of each node is set to 100 meters over 1 Mbps wireless channel.

- Nodes have a mobile pattern which follows the random waypoint model. During the simulation, each node starts moving from its initial position to a random destination inside the simulation area. When it reaches its destination, it pauses for some time, then continues its movement again, and so on.

### 6.2 Energy Aware Dynamic Task allocation Simulation Results

In all experiments, we assume that all nodes are processing nodes with different processing capabilities based on the node's processing id number which is selected randomly at the beginning of each experiment. We assume that each node is given a couple of the Two-Dimensional Array (TDA), one is the processing time and the other is the energy consumption, which are taken for all task types on all node types. Tasks are generated on each node according to the Poisson Process. Each node has an energy level that is gradually consumed when the node executes a task or when it sends (or receives) a message. The simulation model consists of 70 mobile nodes which are distributed randomly in a 700x700 square meters area.

### 6.2.1 Performance Metrics

Three methods of evaluation are used to assess our scheme's performance: the first one is by evaluating the *scheduling efficiency* which is measured two metrics:

- Average Delay which is the average consumed time from generating the task until the time at which the task is submitted to be processed.
- Average Energy Consumption which is the average amount of the energy consumed by each task.

While the second is the *network lifetime* (the time until the first node dies), which is determined by calculating the minimum node energy in each experiment. The last method is the *network scalability* which is measured by computing the task's communication energy against the arrival rate in each experiment. At the end, the obtained results are compared with those of model in [16].

### 6.2.2 Ad Hoc Network characteristics

We use some abbreviations to describe the following characteristics of the network.

- Node Mobility (pause time) – P: defines the pause time.
- Transmission Range -TR: the maximum distance at which the data transmitted by a node can be heard.
- Number of nodes – N: the total number of nodes.
- Client Nodes – CN: the nodes at which the tasks are generated.
- Task arrival rate – $\lambda$ : the rate by which the tasks arrive.
- Channel Capacity CC: the data transmission rate.

### 6.2.3 Comparison study between EADSSTA and EADTA

In this section we compare EADSSTA model to EADTA model in term of the average delay, the consumption of energy, the network lifetime, and the network scalability.

#### 6.2.3.1 The Effect of Arrival Rate

The main concern of this experiment is to test how the arrival rate affects the performance of EADSSTA and EADTA. Tasks are generated by 30 nodes with a task arrival rate of 1-5 tasks/node/sec. whereas, the other nodes are used to execute the remote tasks based on their states, i.e. whether a node is a server or not. The *pause time* is set to be 100 sec, and the movement speed of each node is 0-2m/sec. This experiment is actually made to understand how the arrival rate affects: (1) average delay, (2) consumption of energy, (3) network lifetime, and (4) network scalability.

Fig. 6.1(a) shows the effect of the arrival rate on the average delay in both methods, EADTA and EADSSTA. Upon increasing the arrival rate, more tasks are needed to be allocated, and a longer waiting queue at each node will then come about. This will in turn result in an increase in the time delay in EADTA as well as EADSSTA. After point 3, EADTA continues in its rapid increase, as opposed to EADSSTA, which increases slightly. Therefore, EADSSTA incurs better performance with the load increase.

**P=100, TR=100, CN=30, N=70, CC=1**

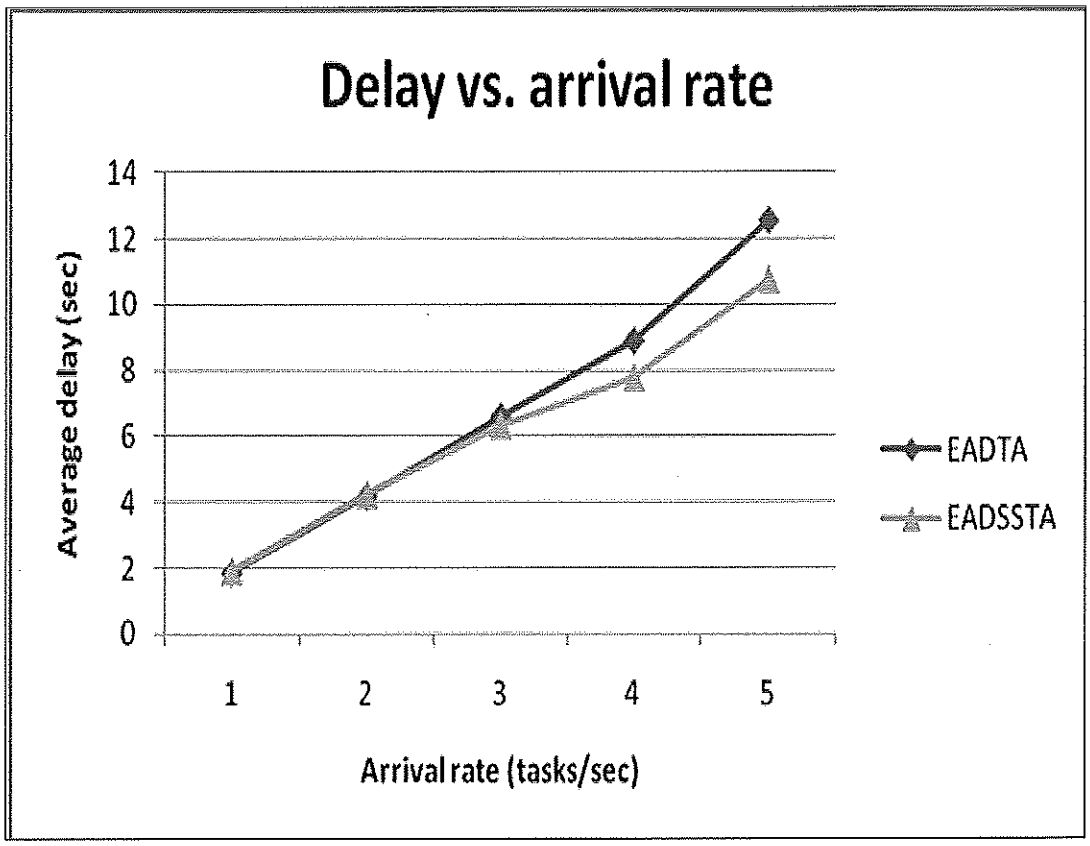



Figure 6.1(a): Average task delay vs. arrival rate

In regards to the energy consumption, Fig. 6.1(b) illustrates that the energy consumption of EADTA increases sharply with the increase of the arrival rate, where it increases slowly in EADSSTA. Therefore it can also be concluded that EADSSTA is more efficient than EADTA in terms of energy efficiency, as well.

P=100, TR=100, CN=30, N=70, CC=1

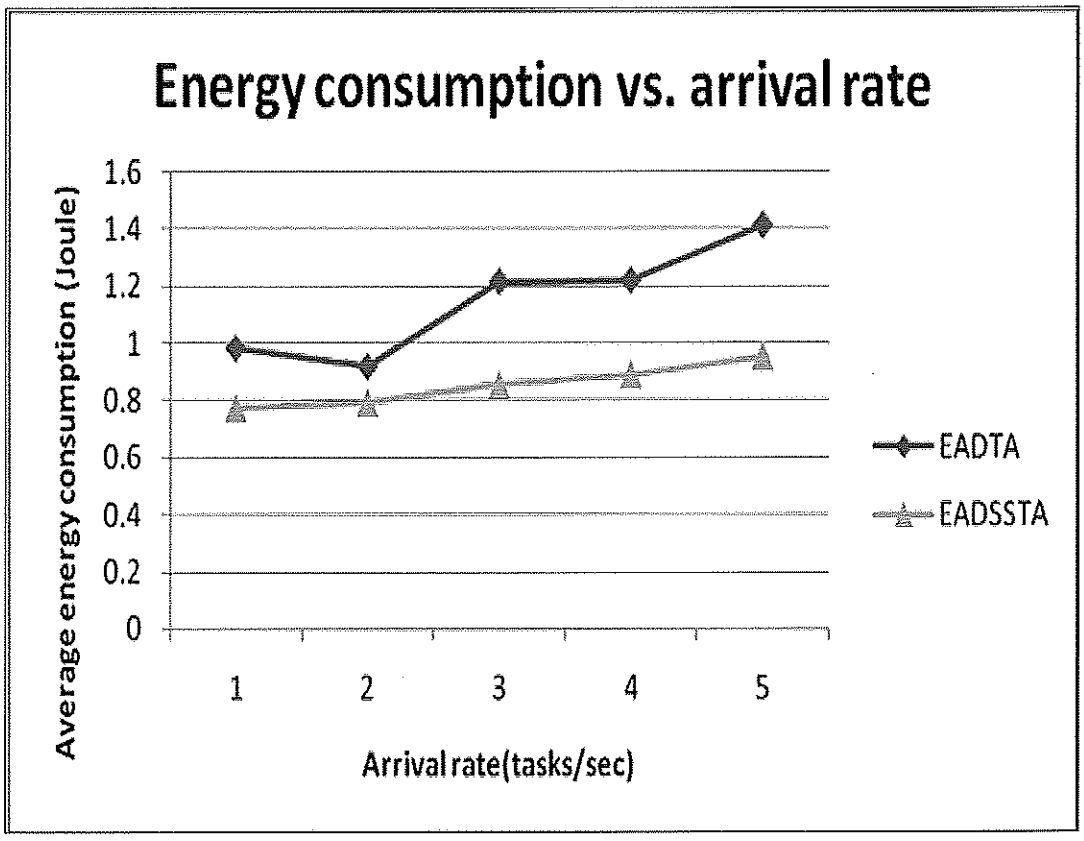



Figure 6.1(b): Energy Consumption vs. arrival rate

Fig 6.1(c) shows that the minimum node energy declines as the arrival rate is increased. This means that the number of the remaining living nodes decreases, leading to a shortage in the network lifetime. This can be seen in both methods EADTA and EADSSTA, however, as can be seen, EADTA is more sensitive to this effect than EADSSTA. Therefore, the lifetime in EADSSTA is longer than that of EADTA.

**P=100, TR=100, CN=30, N=70, CC=1**

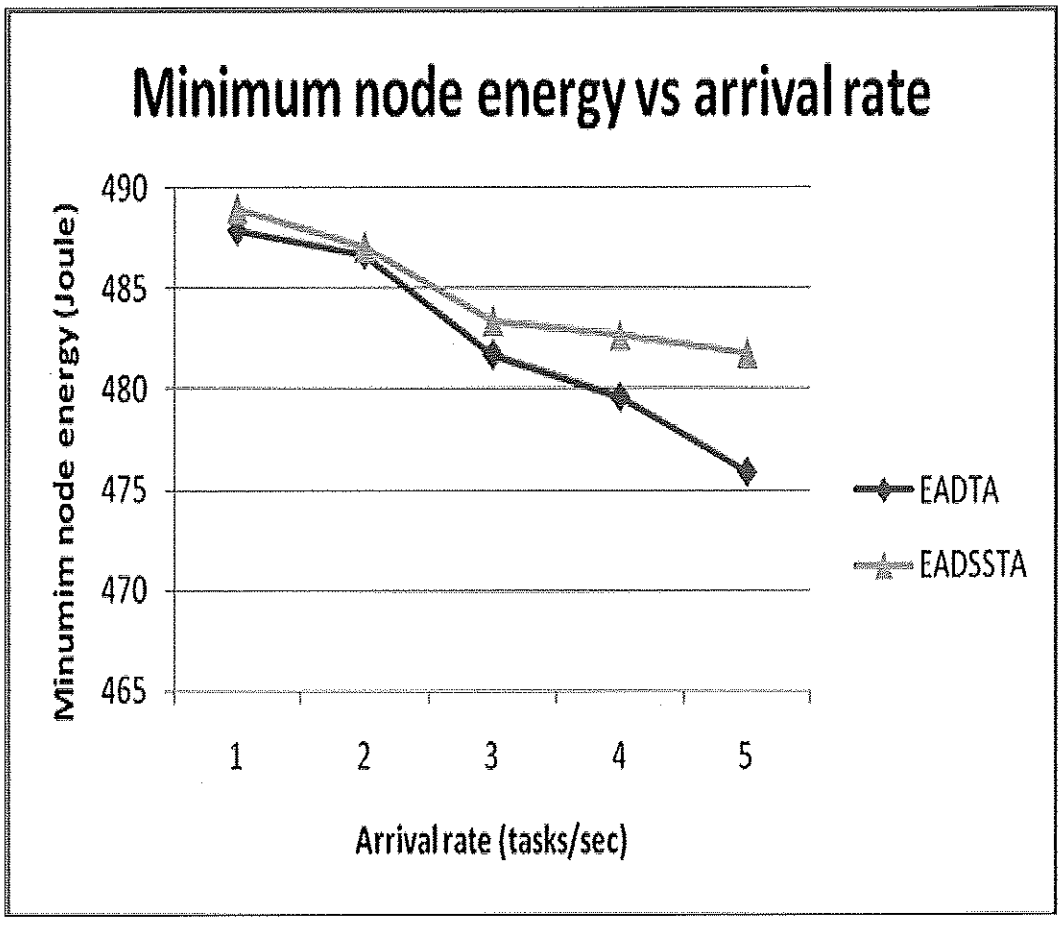



Figure 6.1(c): Minimum node energy vs. arrival rate

As for the communication energy consumption, it significantly increases when the load is increased. This can be observed from Fig 6.1(d), which shows that, when EADTA is used, the communication energy consumption increases sharply as the arrival rate is increased. On the other hand, in the case of EADSSTA, the effect is smaller and the increase rate is less. For example, at point 5, EADTA incurs about %30 in energy consumption more than in EADSSTA.

P=100, TR=100, CN=30, N=70, CC=1

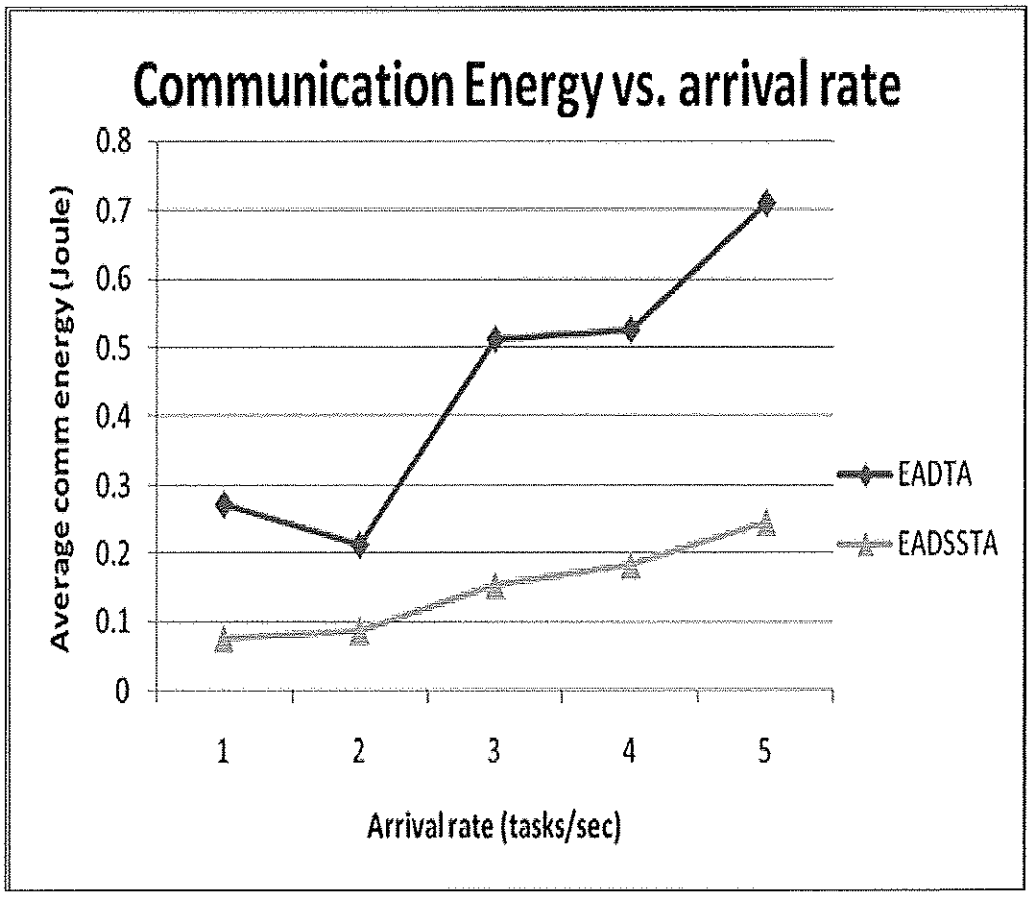



Figure 6.1(d): Communication energy vs. arrival rate

### 6.2.3.2 The Effect of Mobility

In the second experiment, we examine the effect of mobility on the performance in both EADSSTA and EADTA. Here, the parameters are:

- Number of client nodes is 30
- Arrival rate is 4 tasks/sec in each client node
- Pause times are 20, 40, 60, 80, and 100

When comes to mobility, Fig. 6.2(a) shows that the delay in EADSSTA increases with the increase in pause time, however, at point 60, the delay reaches to the lowest point, this can be explained that the 60 seconds pause time is the best point at which nodes are more stable and the EADSSTA can allocate the remote tasks to the best servers. When the pause time is short, the best servers can not be reached or far away, therefore, the allocation is poor. From Fig 6.2(a) we can see that the best point of EADTA is 80 seconds which means that EADSSTA is faster than EADTA. After the best points, both EADSSTA and EADTA incurs high delay, this can be explained that when the pause time is long, nodes are distributed evenly, therefore, the communication among nodes is high which leads to increase the delay. Due to the few communication messages used by EADSSTA, it outperforms EADTA in term of delay in all points.

**λ=4, TR=100, CN=30, N=70, CC=1**

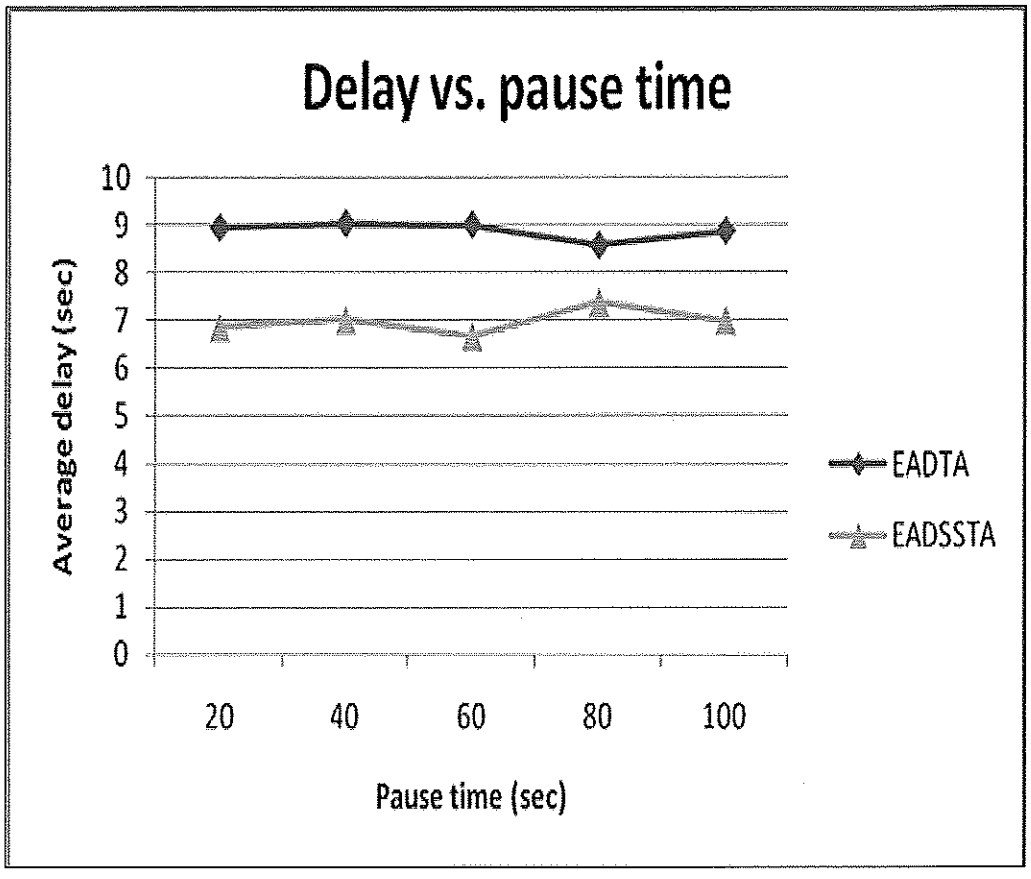



Figure 6.2(a): Average task delay vs. pause time

In Fig 6.2(b) we can see that energy consumption in EADTA increases with the increasing in pause time, however, it goes down in the point 80 which is corresponding to the delay trends, therefore, the same explanation of the delay is also valid here. The mobility has small effects on energy consumption in case of EADSSTA; this is as result of the technique which is renewing at the beginning of each round.

**λ=4, TR=100, CN=30, N=70, CC=1**

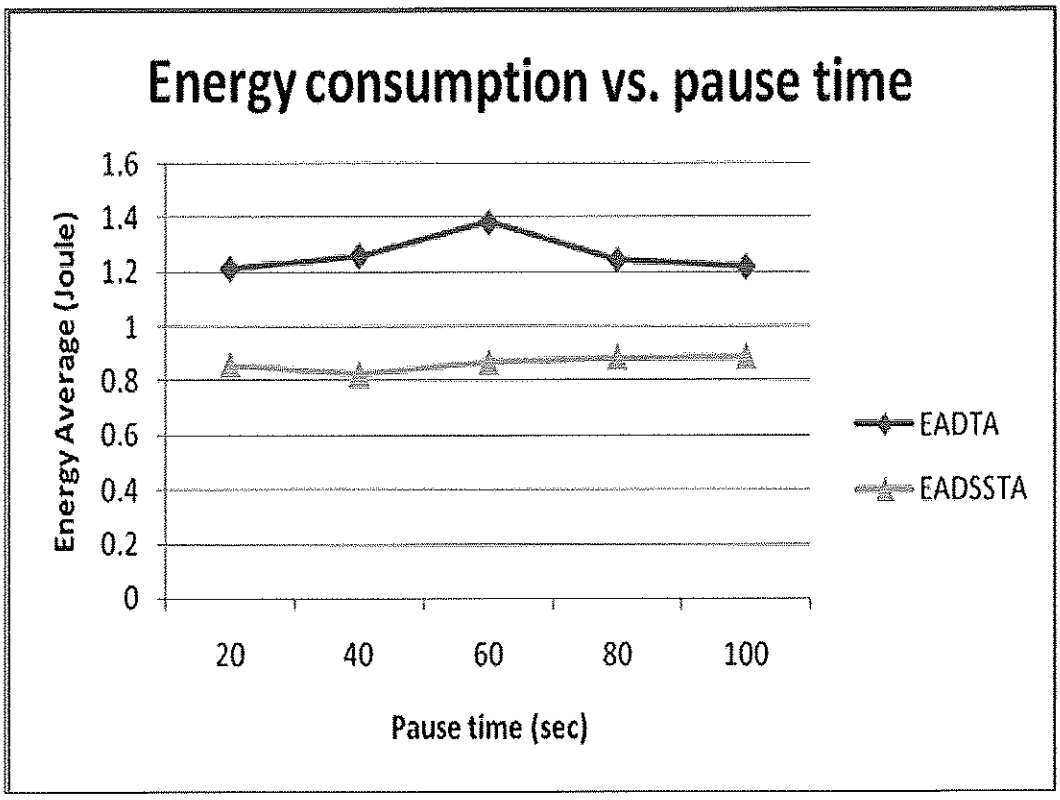



Figure 6.2(b): Consumption energy vs. pause time

Fig. 6.2(c) relates to the effect of the pause time on the node energy consumption. It shows that as the former increases the energy consumption decreases, due to the decrease in the frequency of change in the topology. As a result, the load is distributed among many servers. As can be seen from Fig. 6.2(c), the decrease in the node energy continues until point 80, after which the node energy declines due to isolation and/or congestion.

**λ=4, TR=100, CN=30, N=70, CC=1**

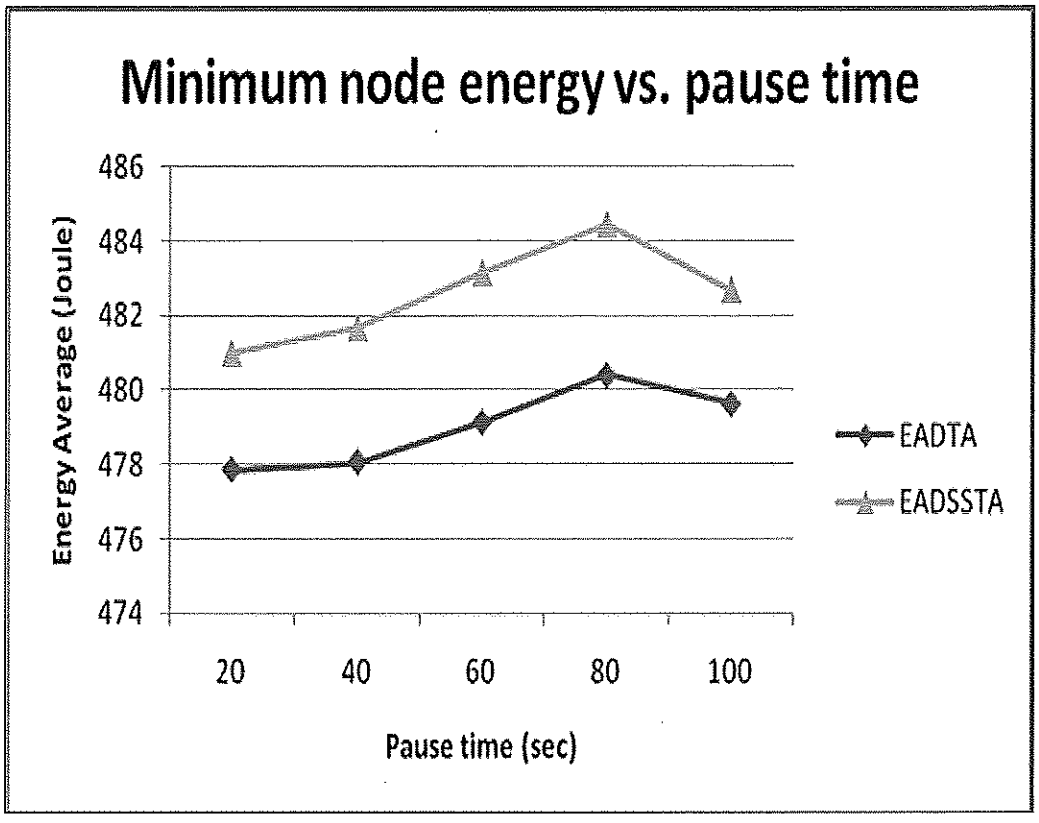



Figure 6.2(c): Minimum node energy vs. pause time

Mobility has a slight effect on the communication energy consumption in case of EADSSTA, as mentioned before, this is due to the updating of the severs over time. Yet, Fig. 6.2(d) shows that, in case of EADTA, the communication energy increases until point 80 at which the energy goes down. This can be explained, as at 80 the nodes start to be stable, the best node can be reached, in addition that the breaking in routing decreases.

**λ=4, TR=100, CN=30, N=70, CC=1**

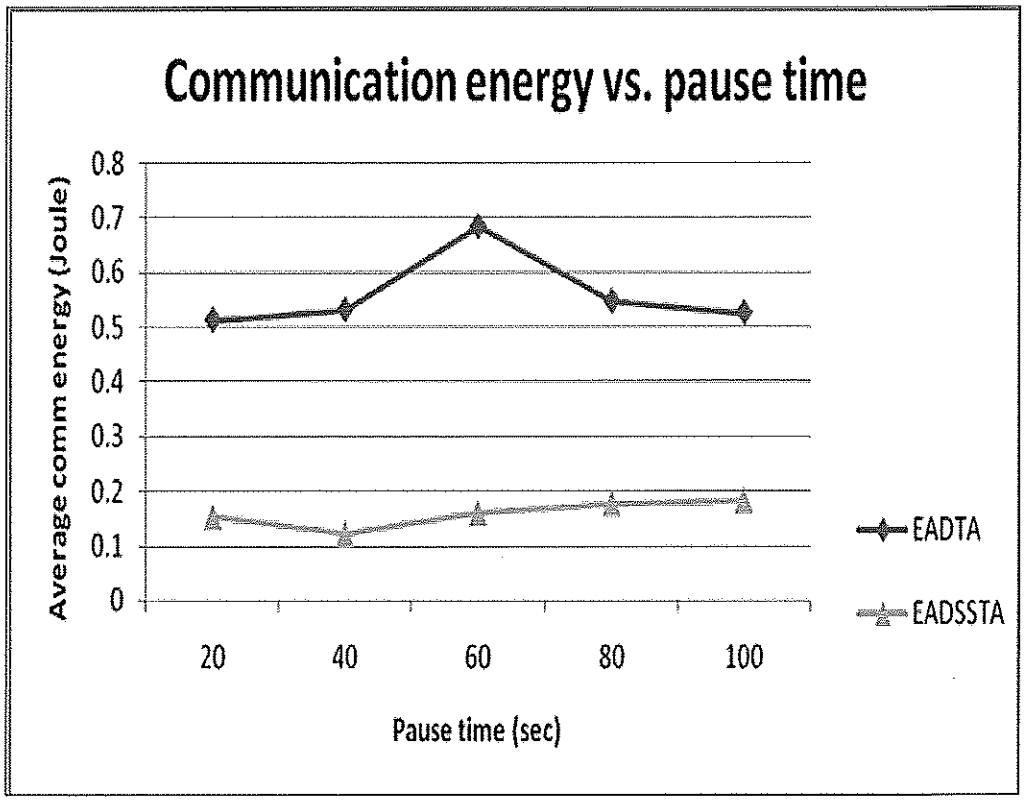



Figure 6.2(d): Communication energy vs. pause time

### 6.2.4 Comparison study between SEADTAS and EADTA

In this section we compare SEADTAS model to EADTA model in term of the average delay, the consumption of energy, the network lifetime, and the network scalability.

#### 6.2.4.1 The Effect of Arrival Rate

This experiment presents a comparison between SEADTAS and EADTA. Tasks are generated by 24 nodes with a task arrival rate of 1-5 tasks/node/sec. The other nodes are used to execute the remote tasks based on their states, i.e. whether a node is a parent or not. The *pause time* is set to be 100 sec, and the movement speed of each node is 0-2m/sec. The main concern of this experiment is to understand how the arrival rate affects: (1) average delay, (2) consumption energy, (3) network lifetime, and (4) network scalability.

Fig. 6.3(a) makes an analogical illustration of the average delay results, obtained by SEADTA and EATDA. Upon increasing the arrival rate, more tasks need to be allocated, the fact that causes an increase in the length of the waiting queue, which in turn, incurs more delay and in both approaches. The average delay in EADTA increases more rapidly than those of SEADTA. This difference in the increase becomes more significant when the arrival rate exceeds 3 tasks/sec at which the gap between the two curves increases. Two factors make SEADTA outperforms EATDA in terms of delay. The first one is that SEADTA takes both current and estimation loads of the parents into account when it allocates a remote task, the second is the fast remote allocation, which in some cases requires only sending one request message to the parent.

P=100, TR=100, CN=24, N=70, CC=1

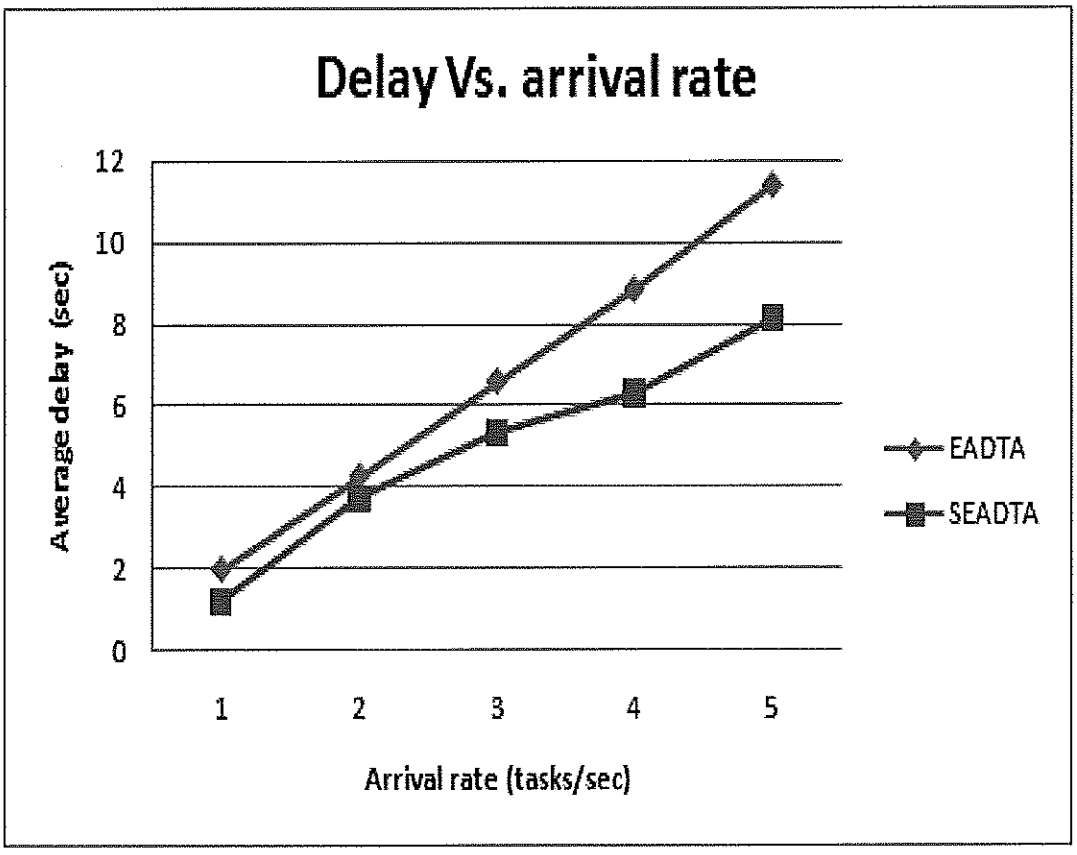



Figure 6.3(a): Average task delay vs. arrival rate

Fig. 6.3(b) illustrates the energy consumption results of SEADTA and EATDA. As mentioned before, upon increasing the arrival rate, more tasks need to be allocated, the fact that causes an increase in the length of the waiting queue, which leads to the increase in the remote tasks. Since EADTA inquires all nodes in the network, the collision of packets increases which results in maximizing the energy consumption. This fact is showed by Fig. 6.3(b).

P=100, TR=100, CN=24, N=70, CC=1

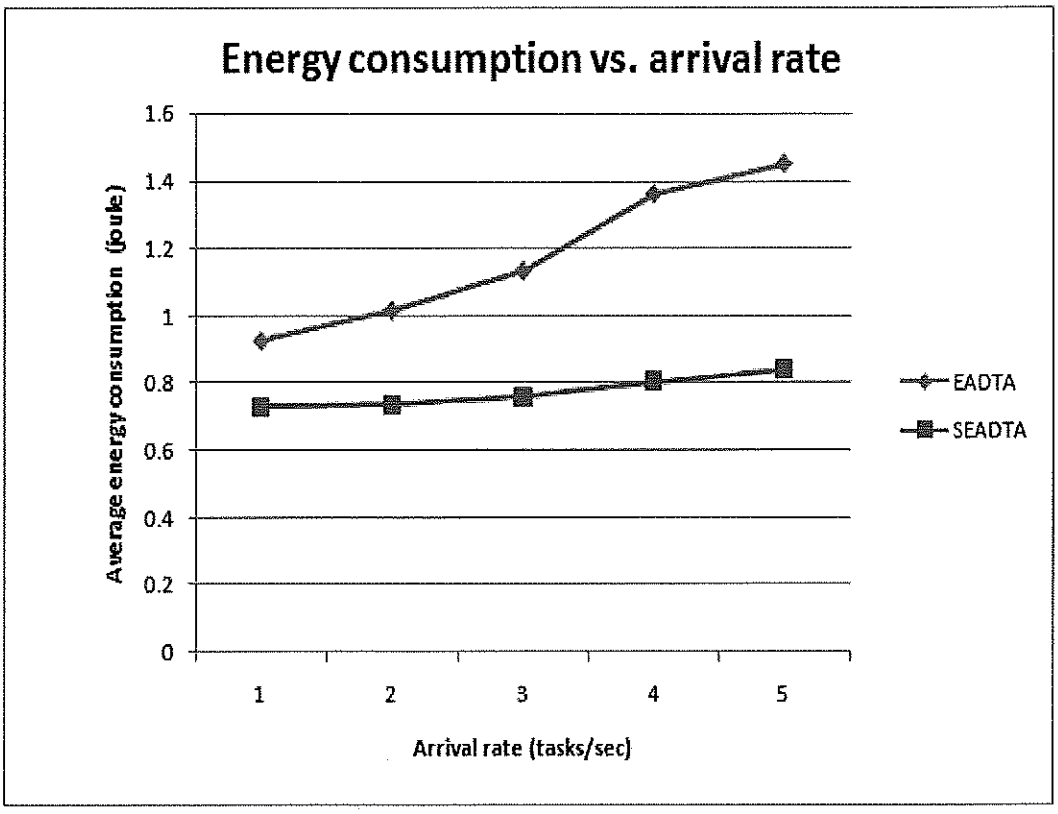



Figure 6.3(b): Energy consumption vs. arrival rate

Moreover, as can be seen in Fig. 6.3(c), the energy of nodes in EADTA decrease more rapidly than those in SEADTA, subsequently declining the number of nodes that will remain alive. SEADTA not only selects the highest-energy nodes to work as parents to receive the remote loads but also it renews them over the time in order to distribute the remote load among many nodes in the network, therefore, SEADTA proves a better performance in terms of energy consumption.

P=100, TR=100, CN=24, N=70, CC=1

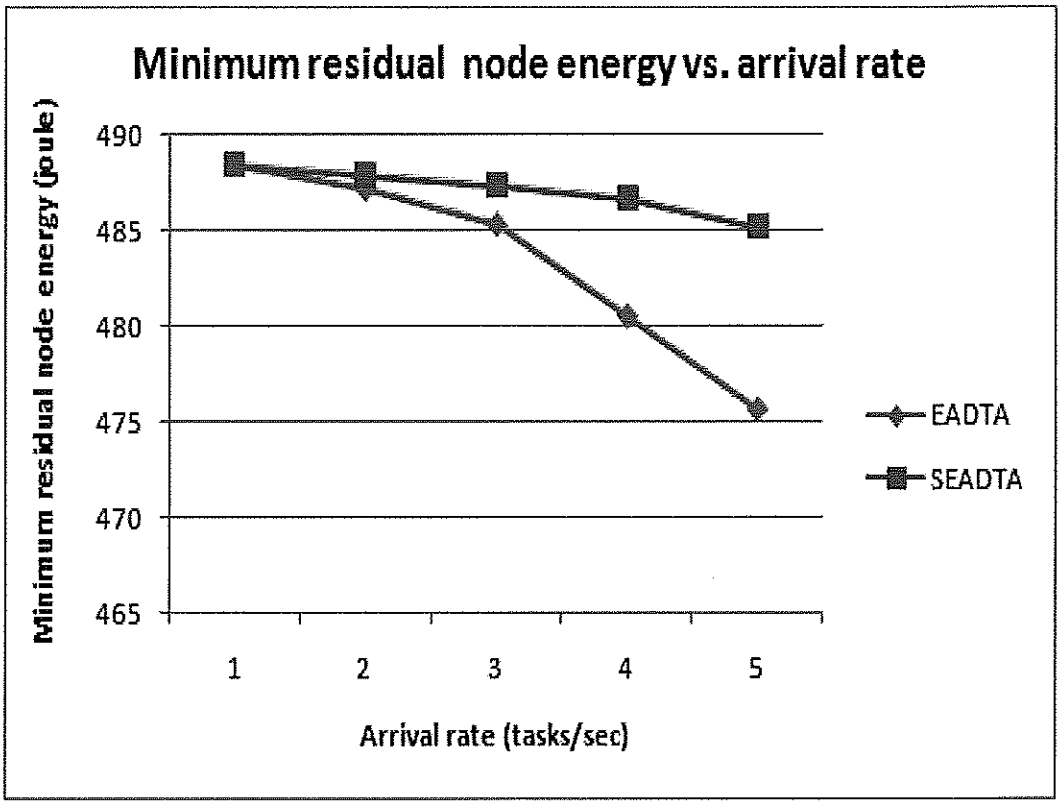



Figure 6.3(c): Minimum residual energy node vs. arrival rate

Figure 6.3(d) shows how EADTA's Comm. Energy increases faster than that of SEADTA, resulting in a reduction in the network lifetime as the arrival rate is increased. The remote allocation in SEADTA may require one message if the sender node has a parent and it accepts to execute the remote task, otherwise the node will inquire the other parents (which are less than total number of nodes) in the network. In both cases the number of messages is less than those in the case of EADTA. Therefore, it can be concluded that the performance of SEADTA is better than that of EADTA.

P=100, TR=100, CN=24, N=70, CC=1

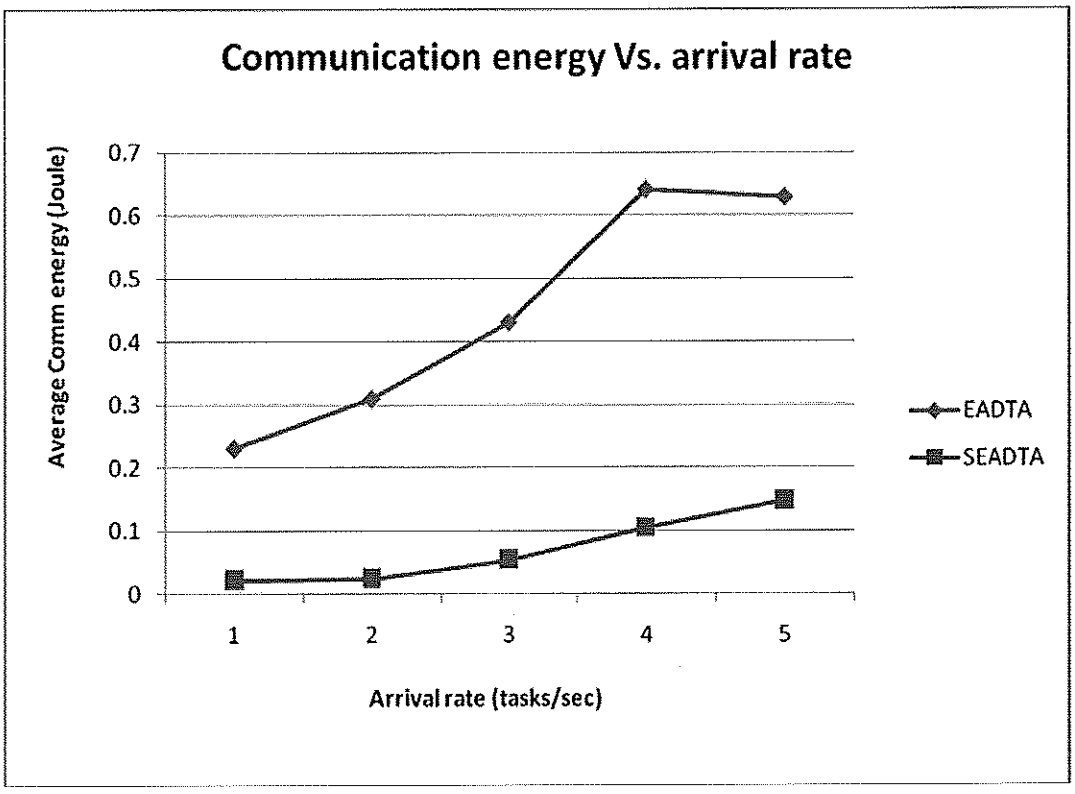



Figure 6.3(d): Communication Energy vs. arrival rate

### 6.2.4.2 The effect of Mobility

In this experiment, we examine the effect of mobility on the performance in both SEADTA and EADTA. Here, the parameters are:

- Number of processing nodes is 30.
- Arrival rate is 4 tasks/sec.
- Pause times are 20, 40, 60, and 100.

The delay of SEADTA increases with the increase in pause time, however, at point 60 the delay reaches to the lowest point, this means that the 60 second is the best point of SEADTA, and the nodes are stable and the best parents are accessible which result in a good remote allocation. The same explanations of EADSSTA and EADTA in Fig. 6.2(a) are valid for SEATA and EADTA in Fig. 6.4(a) respectively.

**λ=4, TR=100, CN=30, N=70, CC=1**

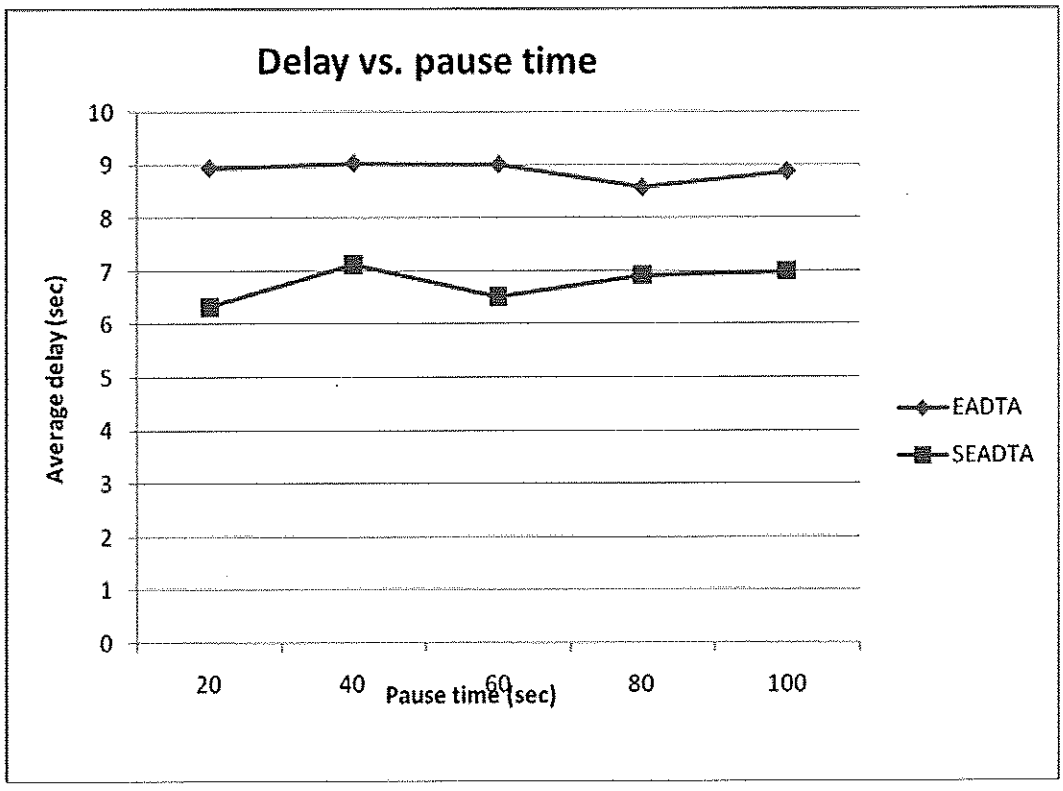



Figure 6.4(a): Average task delay v. pause time

On the other hand, as can be seen in Fig. 6.4(b), the mobility has no effect on the energy consumption of SEADTA. The point 60 is the worst point for EADTA at which the nodes are not stable and the best nodes could not be reached, as a result the remote allocation is poor.

λ=4, TR=100, CN=30, N=70, CC=1

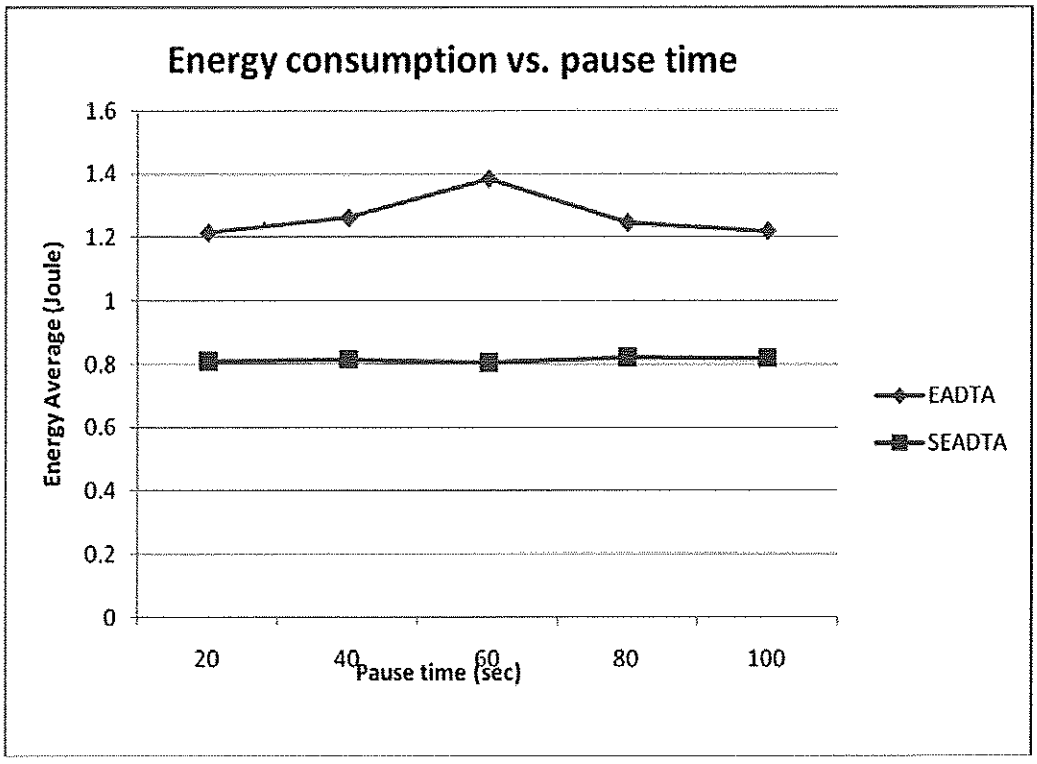



Figure 6.4(b): Energy consumption vs. pause time

In Fig 6.4(c) we can see that the network lifetime of SEADTA is increased in 20 and 100. At point 20 seconds, nodes move fast and parents are renewed in short intervals of time, therefore, the load is distributed among many nodes. At point 100 nodes are almost do not move and the best parents can be reached. In case of EADTA, with the increase in pause time, nodes become more stable and the remote allocation also improves, therefore, the energy of nodes increase, however, when the pause time is long, the communication time increases, as result, the energy decreases.

λ=4, TR=100, CN=30, N=70, CC=1

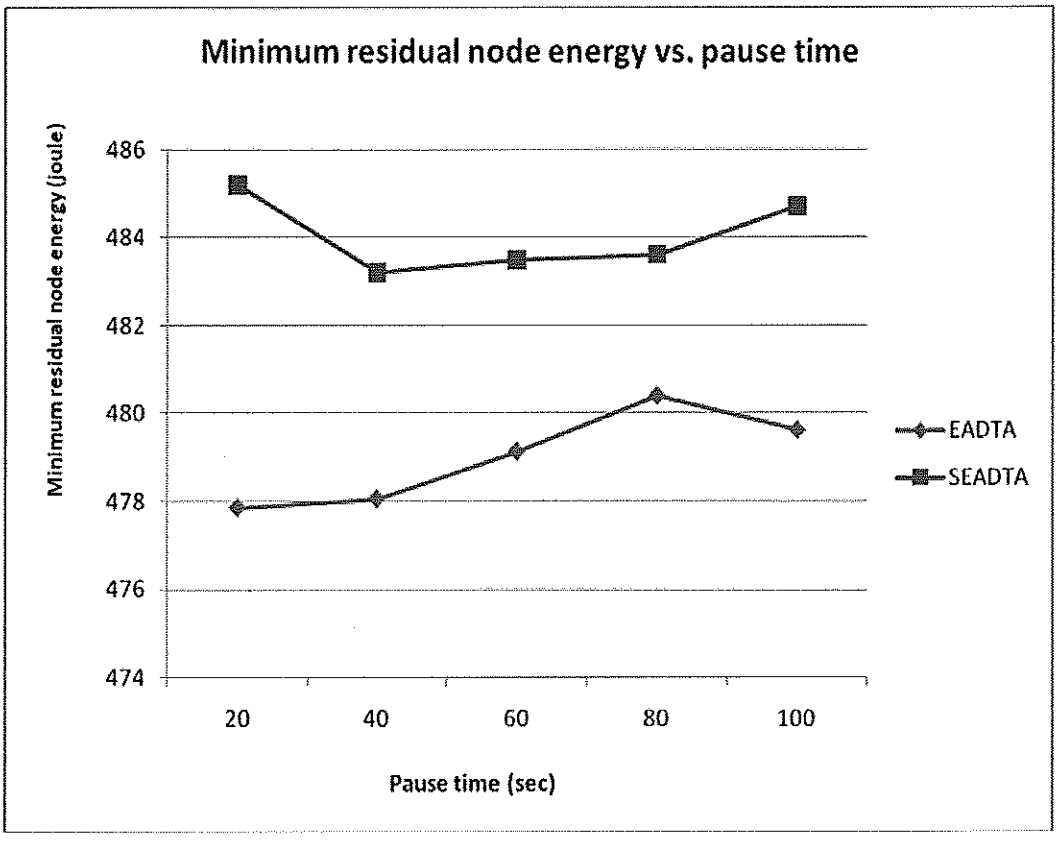



Figure 6.4(c): Minimum node energy vs. pause time

As for the communication energy, we can see from Fig. 6.4(d) that SEADTA is affected by the mobility; this is because SEADTA adapts its behavior according to the changes in the topology by increasing/decreasing the time interval of renewing parents. However, when the pause time is long, the communication increases, therefore, SEADTA incurs high delay at point 100 seconds. As for EADTA, the explanation of Fig 6.4(b) is also correct here.

**λ=4, TR=100, CN=30, N=70, CC=1**

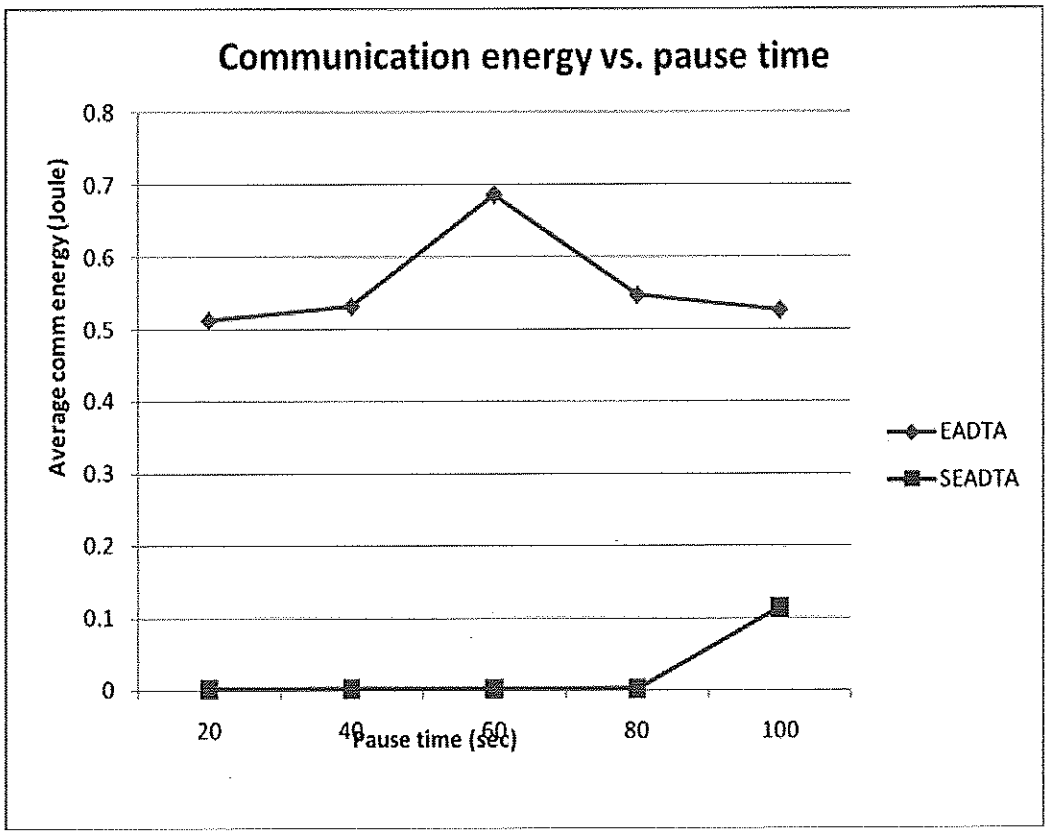



Figure 6.4(d): Communication energy vs. pause time

### 6.2.5 Comparison study between EADSSTA, SEADTAS and EADTA

The objective of this experiment is to compare the three models together in terms of task delay time, energy consumption, minimum node residual energy, and communication energy. The parameters of this experiment are same as in experiment in Section 6.2.3.1.

Fig 6.5(a) shows that both EADSSTA and SEADTA substantially reduce the task delay especially after arrival rate of 3. This is a result of small communication messages used by EADSSTA and SEADTA in scheduling the remote tasks. However, while SEADTA improves task delay by 17%, EADSSTA improves it by 4% as compared with EADTA.

**P=100, TR=100, CN=30, N=70, CC=1**

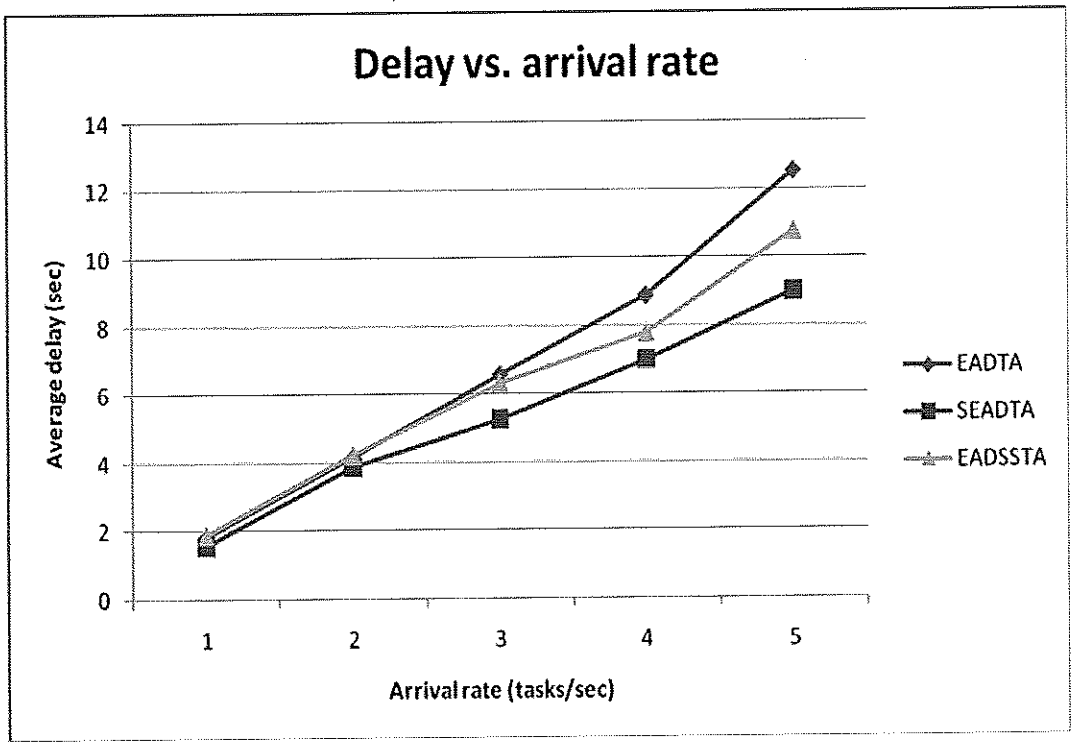



Figure 6.5(a): Average task delay v. arrival rate

Fig 6.5(b) reveals that the energy consumption is decreased by 25% in case of EADSSTA, whereas SEADTA reduces it %30. This result explains the

benefit of the proposed topology managements schemes (DSS and CS) and proves that both EADSSTA and SEADTA are efficient in terms of energy consumption.

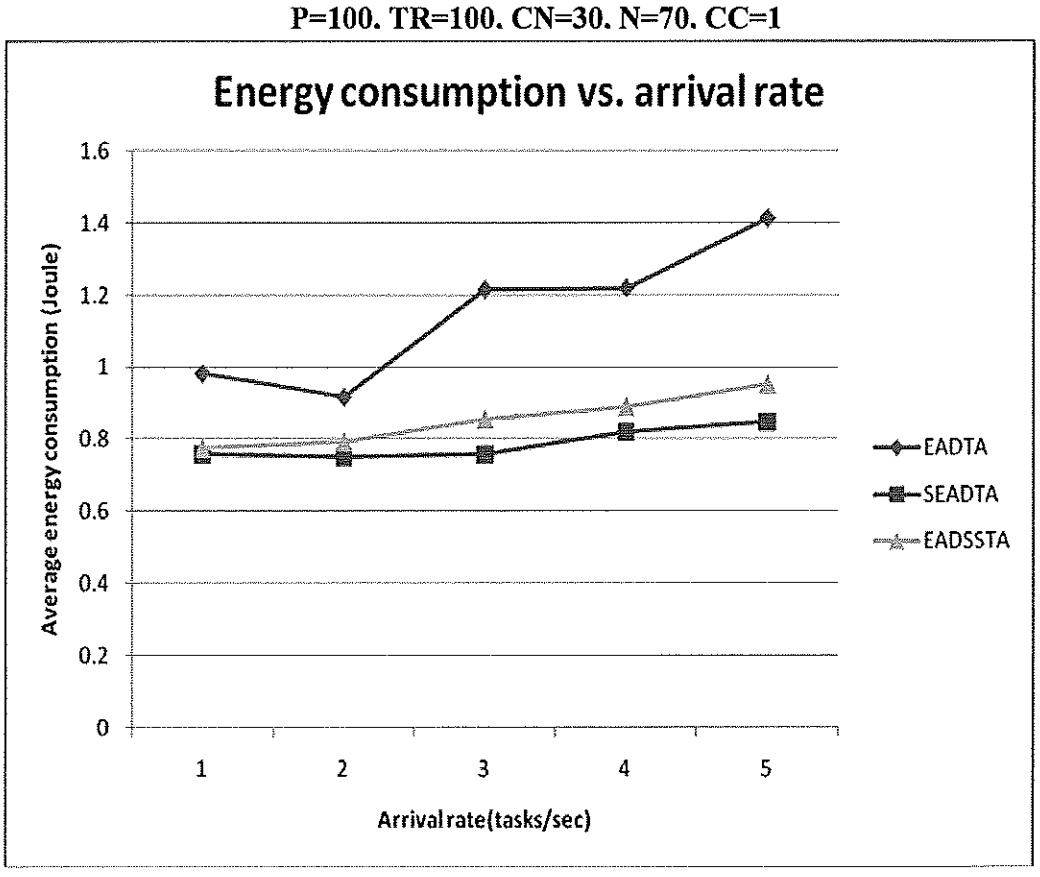



Figure 6.5(b): Energy consumption vs. arrival rate

The minimum residual node energy in fig 6.5(c) shows the network lifetime in each model. As can be seen from the plot, when the number of tasks increases, the node energy in EADTA decreases rapidly, on the other hand, the decreasing in EADSSTA is slow, which is due that EADSSTA inquiring only the servers in the network. However, SEADTA incurs the smallest decrease, which is a result of the network structure used by SEADTA.

P=100, TR=100, CN=30, N=70, CC=1

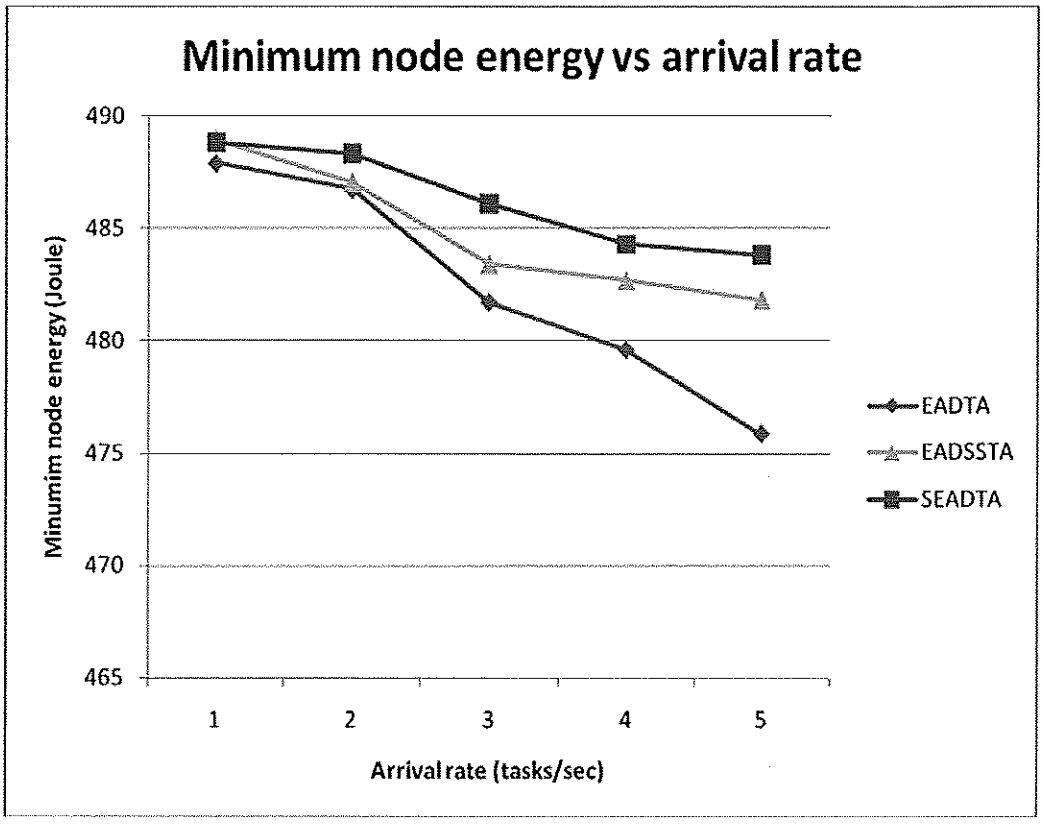



Figure 6.5(c): Minimum node energy vs. arrival rate

Fig 6.5(d) shows that EADTA incur high communication energy. This observation can be explained as a result of inquiring all nodes in the network, which is used by EADTA. In case of EADTA, the communication energy consumption increases rapidly, when arrival rate increases. The increase in the communication energy consumption in EADSSTA and SADTA is slow, which means that EADSSTA and SEADTA actually outperform EADTA in terms of network lifetime and scalability.

Furthermore, Fig 6.5(d) shows that in terms of communication energy SEADTA achieved about 82% better than EADTA, on the other hand,

EADSSTA achieved about 77% outperformance. This result proves that our model is suitable for the application that incurs a huge communications.

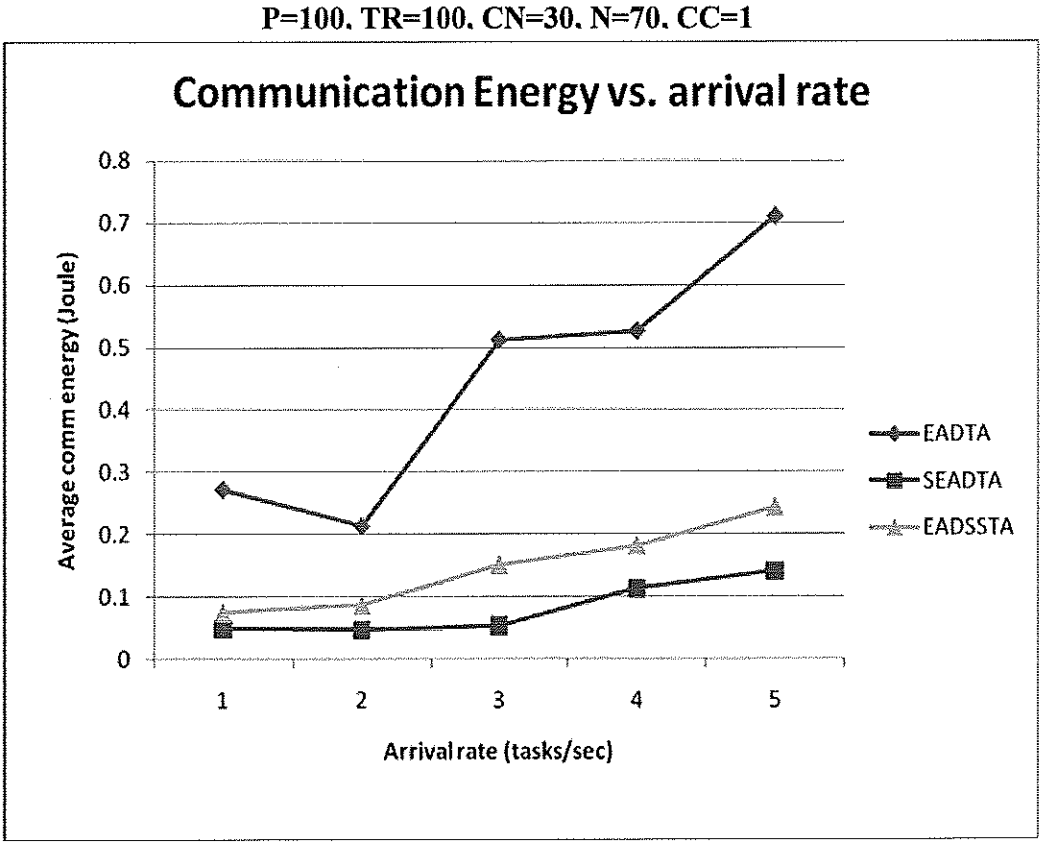



Figure 6.5(d): Communication energy vs. arrival rate

## 6.3 Energy Aware Static Task Scheduling Simulation Results

The objective of ETS model is to schedule tasks in order to minimize the total time and energy. ETS proposes a general energy aware task scheduling method; however, it does not consider the method by which a node collects information about the other nodes in the system. Therefore, we use this model to compare it with proposed models. The simulation model consists of 100 mobile nodes which are distributed randomly inside a 750x750 square meters area. In all of the following experiments, the task and processor types are to ten and six, respectively. The execution energy and time of all task types on all processor types are selected randomly between (0.06 -0.09) joule, and (2-9) μs, respectively. The value of both $\alpha$ and $\beta$ is equal to one to reflect the same weight for time and energy in the cost function. The communication energy between any two neighbored nodes is 5J/byte, while the communication delay which includes medium access, queuing, and transmission delay is 5μs/byte. The size of exchange data between any task and its immediate successor task is selected randomly between 900-1000 byte.

The simulation time is divided into two equal rounds. At the beginning of each round the nodes are allowed some time to move inside the simulation area, and when they stop their movement, CS or DSS are executed to form new clusters or select new servers.

### 6.3.1 Simulation Results

We have conducted three main experiments to assess the proposed models. In each experiment, we compare the studied models in terms of the makespan and the minimum remaining node energy. To compare makespan values, we conduct an experiment that generates task graph with 40 tasks on 4 random nodes in each round. At the end of the experiment we compute the

average makespan. This experiment is repeated for task graphs with 80, 120, 160, and 180 tasks. As for the energy values we compare the minimum node energy by generating a task graph with 200 tasks and 310 intercommunications among tasks in one executing node per round, and we compute the minimum node energy at the end of the experiment. This experiment is repeated for nodes 3, 5, 7, and 9.

1- In the first experiment, our concern is to compare EA-STS-DSS and EA-STS-CS. As can be seen in Fig 6.6(a), EA-STS-CS improves makespan about 75% over EA-STS-DSS. This is due to the fact that the server selection inquires all servers in the network before starting scheduling the tasks. In the clustering technique, however, only one message is needed to send the task graph to the upper parent in the cluster. Moreover, as depicted in Fig 6.6(b), the decrease in the node energy in EA-STS-DSS is more rapid than that in EA-STS-CS. This is indeed a result of the huge communication energy which leads to the decrease in the number of living nodes in the network.

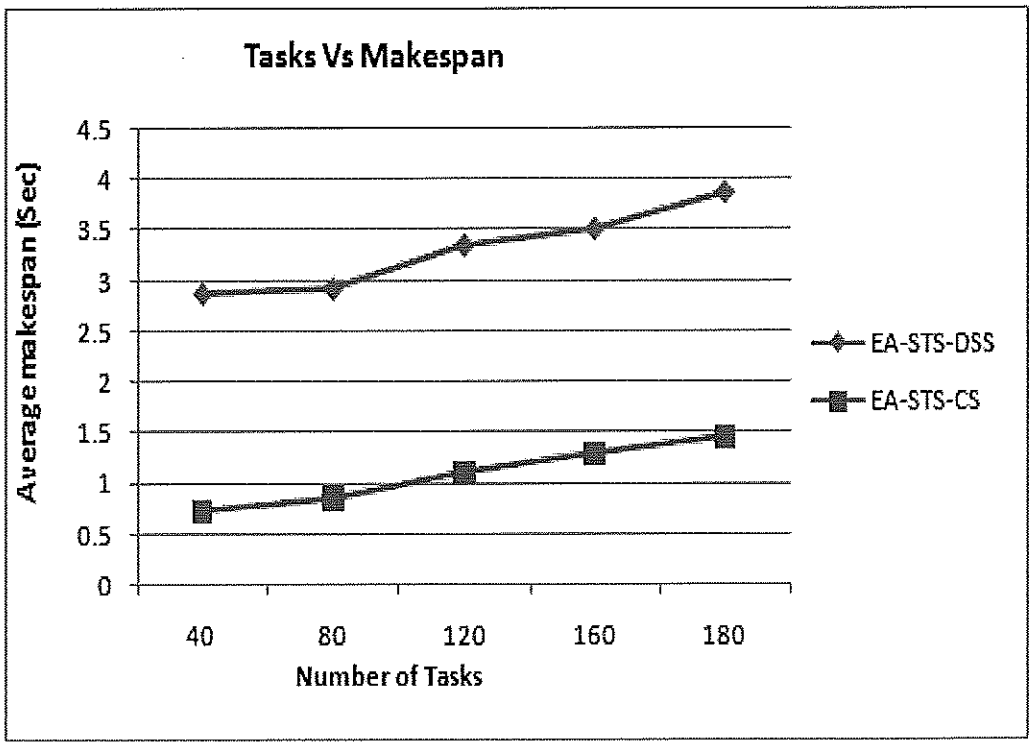



Figure 6.6(a): Average Makespan Vs. Task number

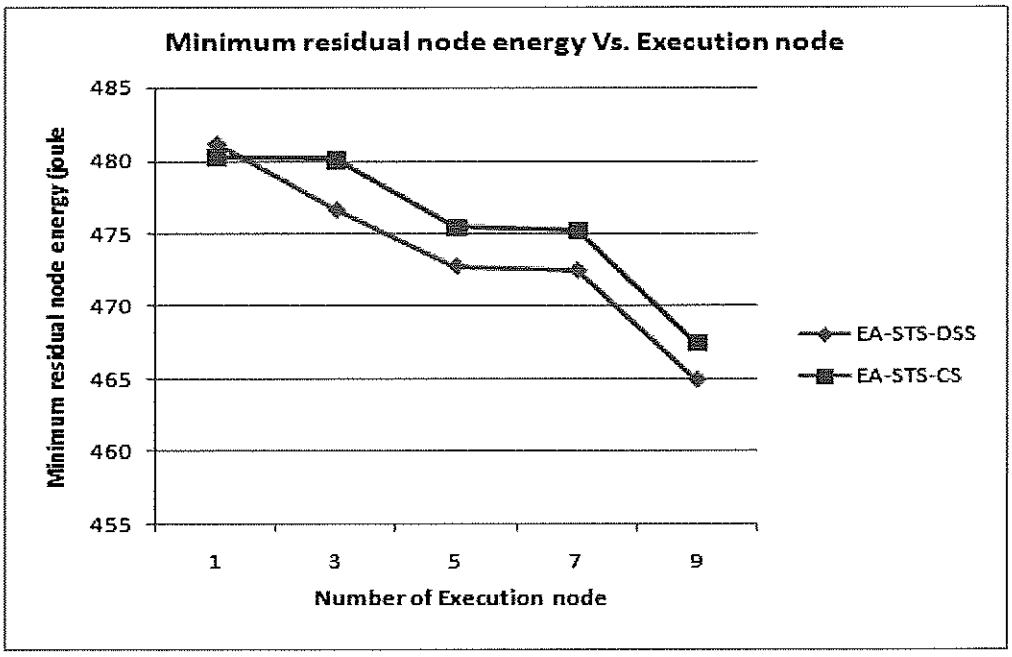



Figure 6.6 (b): Minimum node energy Vs. the number of execution node

2- Fig 6.7(a) and Fig 6.7(b) exhibit the simulation results of the makespan and the minimum residual node energy results respectively, obtained by our model EA-STS-DSS and ETS. As Fig 6.7(a) reveals, EA-STS-DSS makes a smaller improvement in makespan than ETS, This is due to the fact that EA-STS-DSS utilizes the task level technique which is based on the average task executing time rather than the one used by ETS which depends on the minimum executing time of each task. The enhancement of the minimum residual energy, which is achieved by EA-STS-DSS as can be seen in Fig 6.7(b), comes from the fact that EA-STA-SS computes the reaming energy on the node before assigning a task to it, in other words, it tries to assign a task to the node which will maximize the reaming residual energy. This characteristic is absent in ETS, because it only tries to minimize the consumption energy during task scheduling processing.

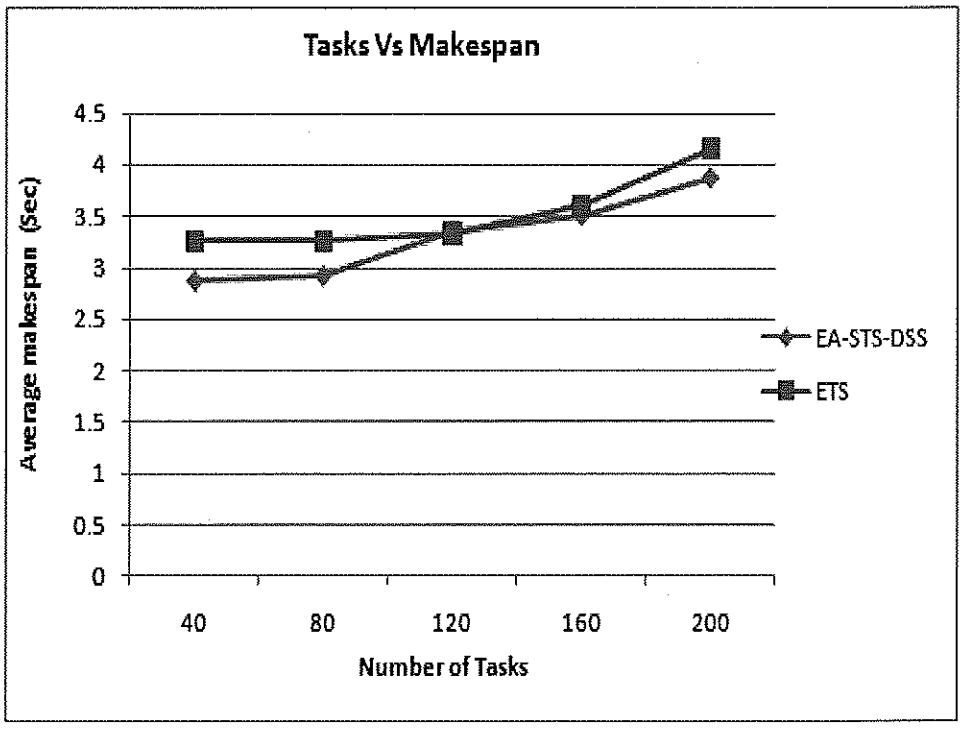



Figure 6.7 (a): Average Makespan Vs. task number

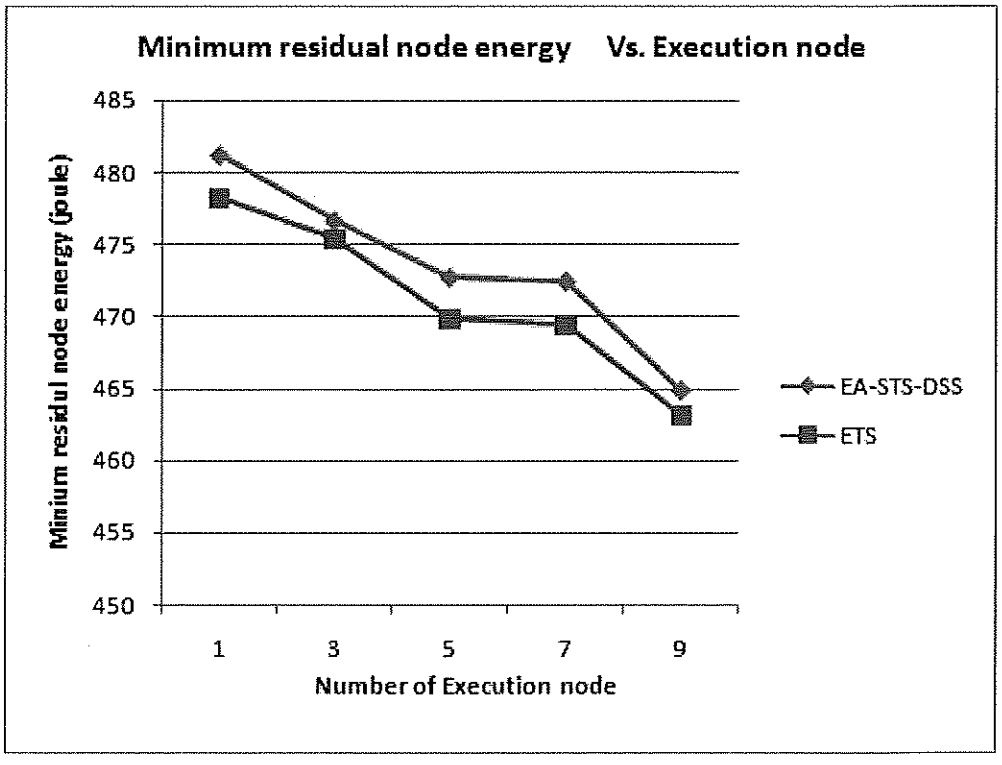



Figure 6.7 (b): Minimum node energy Vs. the number of execution node

3- In the final experiment, we compare EA-STS-CS against the ETS as illustrated in Fig 6.8(a) and Fig 6.8(b). The big gap in the makespan in Fig 4.6(a) reflects the importance of the clustering scheme in reducing the communication time. The same explanation mentioned in experiment 1 (Fig 6.6(a)) is valid also here. Fig 6.8(b) shows that EA-STS-CS makes about 2.5% better saving in the minimum residual energy than ETS. This is in fact a consequence of two main factors: The first one is that EA-STS-CS tries to maximize the reaming energy on all available parents by comparing their reaming energy values, and then it selects the one which has the highest reaming energy. The second factor is that the required communication messages to schedule a graph are only one message sending by the node that has the graph to the upper parent of the its cluster. Conversely, ETS tries to minimize the consumption energy without considering the reaming energy values in the servers.

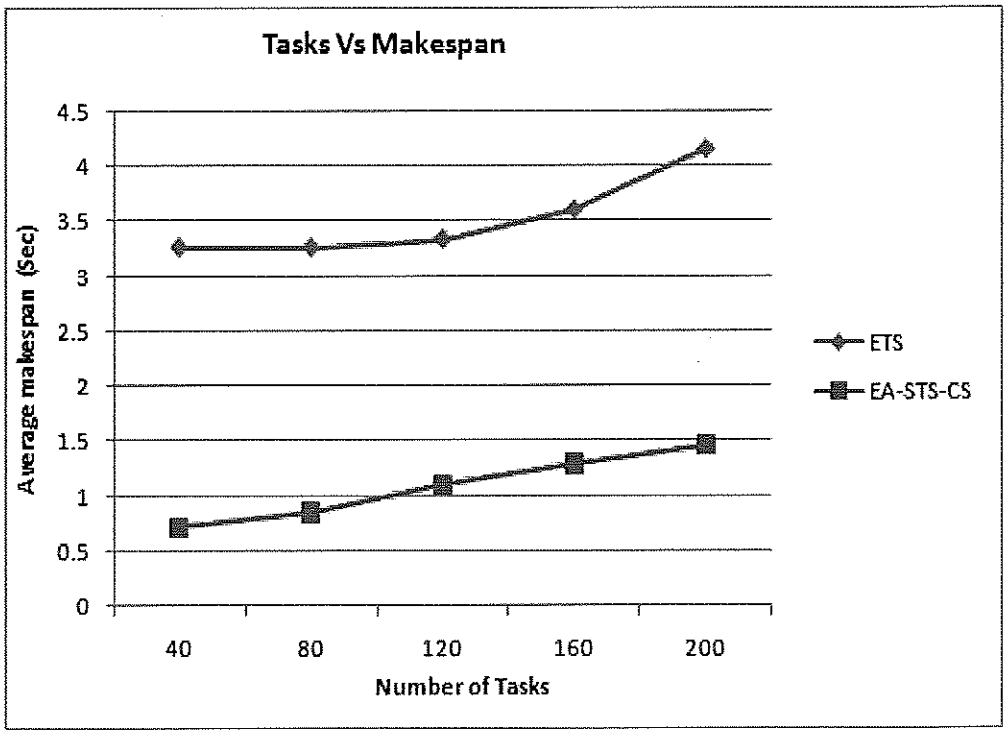



Figure 6.8 (a): Average Makespan Vs. the number of tasks

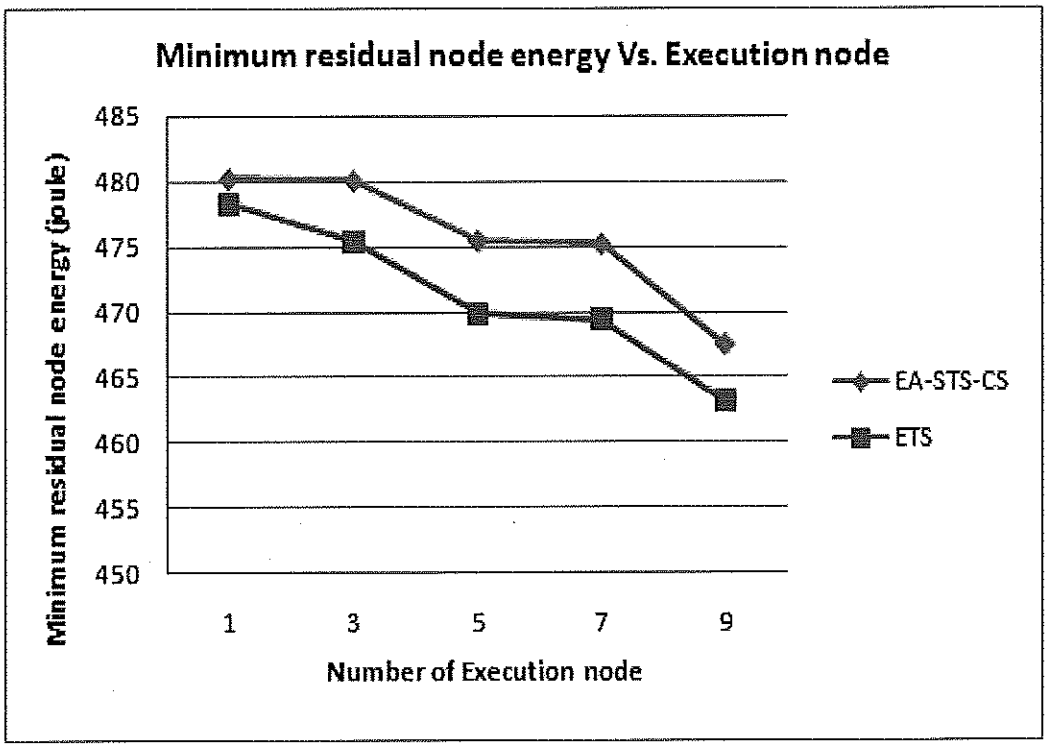



Figure 6.8 ( b): Minimum node energy Vs. the number of execution nodes

# CHAPTER 7

# CONCLUSION AND FUTURE WORK

## 7.1 General

The objective of this thesis is to design a cooperative model for energy aware task scheduling which aims to prolong the network lifetime and make the network more scalable. We introduced two novel techniques for managing the MANET topology and four new task scheduling methods. The first topology management technique is DSS which divides the network nodes into servers and non server nodes. The servers are selected based on the node energy; in fact, the nodes with highest energy are selected to serve as servers at one of the two levels according to their energy levels. The servers at level one have higher energy than those at level two, therefore, they receive more remote load.

The second technique is the clustering scheme (CS) which aims at grouping the nodes into clusters according to their energy levels. Each cluster has four types of nodes: children, one hop parent nodes, intern parents, and the upper parent node. Nodes in each cluster there is one upper parent. The children nodes do not execute any remote tasks whereas the parent nodes execute only the remote tasks provided by their children. The intern parents and the upper parent can execute tasks from inside/outside their clusters.

DSS and CS classify nodes to have different roles. By doing so, these techniques try to direct most of loads to the nodes that have the highest energy in order to prolong the network lifetime.

The formation of DSS and CS is restricted to 3-hope distance. This restriction leads to simplify the management and thus decreases both time and energy required. Furthermore, this design decision agrees with the results of many studies in the literature.

For dynamic task scheduling, this study proposes two energy aware dynamic task allocations methods. The first, EADSSTA, is implemented above DSS. In this model, the node that faces a shortage in its resources (energy or processing capability) inquires the right servers in the network; the servers that are able to execute the task, reply to the node by sending their energy level, their waiting queue length, and some other performance or service metrics. Upon receiving replies, the sender node decides on the best candidate server to be the receiver according to the cost function.

The second technique is SEADTA, in which the concept is that the node that faces shortage in its resources inquires its parent. The parent may not reply, in such case, the method used in EADSSTA is executed except that only the intern parents and the upper parent will reply to the remote call.

For the static scheduling, most of the previous energy aware static scheduling in the literature has not considered the topology management and the method by which the node can collect information about the system. To fill this gap, we proposed two energy aware static scheduling approaches: The EA-STS-DSS is built on DSS. Here the node with the application, inquires the servers in the network, each server replies by sending its energy level and its routing table to the other servers. Upon receiving replies, the node schedules the tasks on the servers that minimize the execution time and maximize the remaining energy base on the proposed cost function.

In the second static task scheduling method, EA-STS-CS, the upper parent in each cluster collects the information about parents and the intern parents immediately after the CS establishment. When a node has a parallel application, it sends only the application (task graph) to its upper parent. Having already collected the information about parents, the upper parent will schedule the tasks over parents and intern parents as in EA-STS-DSS.

To implement the above techniques, the network lifetime is divided into many equal rounds. At the beginning of each round either DSS or CS is executed

to select the new servers or build new clusters. This means that servers and parents are updated over the time to allow the remote load to be distributed among nodes. The cost functions allow selection of the receiver server based on the current load and energy in each candidate server in addition to the required energy to transfer the task. The renewing of the servers and parents at the beginning of each round helps to minimize the side effect of the node mobility.

### 7.2 Concluding Remarks on the Results

Both DSS and CS based topology management reduces the load in the system, as the remote scheduling does not require inquiring all nodes in the network. The experiments showed that the EADSSTA and SEADTA achieved better performance in terms of delay. While SEADTA decreases the delay by 17% over EADTA, EADSSTA decreases it by 4%. This is achieved as a result of the decreasing the number of per- task inquired nodes and the considering the current load in selecting. Compared with EADTA, SEADTA and EADSSTA reduce the energy consumption by 30% and 25% respectively.

The cost function used by the static scheduling assigns a task to the node that will increase the remaining energy. Both task level mechanism and the cost function play important role in decreasing the per task delay.

### 7.3 Future Work

This work can be extended in several directions:

Real time tasks can be studied over the proposed topology models to evaluate the performance of EADSSTA and SEADTA in the real time environment.

It is possible to combine more than one metric, such as energy and processing capability to select the servers or to allow the application to define the metrics by which the servers will be selected.

The length of each round time in the proposed techniques was determined before the simulation starts. It is possible to study this problem to find out a method that determines the length of each round dynamically during the execution time.

In the sensor network platforms, nodes around the base stations consume their energy faster than those that are far from the base stations. One solution to this problem is to have mobile base stations that change their places from time to time. However, this technique needs a method to compute the optimal place of each base station. In this sense, DSS and CS can be extended to define the optimal places of the base stations according the nodes energy.

## REFERENCES


[1] C. Cordeiro, and D. Agrawal, Ad HOC & SENSOR NETWORKS, *World scientific, 2006.*

[2] Z. Zong, M. Nijim, A. Manzanare, and X. Qin, Energy Efficient Scheduling for Parallel application on Mobile Clusters, *Cluster Computing, Volume 11, 2008, 91-113.*

[3] I. Chlamta, M. Conti, J. Jennifer, and N. Liu, Mobile ad hoc networking: imperatives and challenge, *Ad Hoc Networks 1(2003), 13-64.*

[4] S. Singh, M. Woo, and C. Raghavendra, Power-Aware Routing in Mobile Ad Hoc Networks, *in Proceedings of Mobihoc, 1998, 181-190.*

[5] K. Jin, and D. Cho, A MAC Algorithm Energy-Limited Ad Hoc Networks, *in proceedings of fall VTC 2000, September 2000, 219-222.*

[6] S. Agarwal, S. Krishnamurthy, R. Katz, and S. Dao, Distributed power Control in Ad-hoc wireless Networks, *IEEE PIMRC, Vol. 2, pp. F-59-F-66, 2001.*

[7] E. Jung, and N. Vaidya, A Power Control MAC Protocol for Ad Hoc Networks, *in ACM Mobicom, 2002.*

[8] J. Wieselthier, G. Nguyen, and A. Ephremides, On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networkss, *IEEE INFOCOM, March 2000.*

[9] Yu, J.Y. and Chong, P.H.J., A survey of clustering schemes for mobile ad hoc networks, *Communications Surveys & Tutorials, IEEE Volume 7, Issue 1, First Qtr. 2005 Page(s): 32 – 48*

[10] NS2 simulator main web page, http://www.isi.edu/nsnam/ns/, last visited on 15 October 2008.

[11] R. Chow, T. Johnson, Distributed operating systems and algorithms, *Addison Wesley Longman, 1997.*

[12] T. Lewis, H. El-Rewini, Introduction to parallel computing, *prentice-Hall International Editions, 1992.*

[13] C. Lee, J. Hwang, Y. Chow, and F. Anger, Multiprocessor scheduling with interprocessor communication delays, *operations Research Letters, Vol. 7, Number 3, June 1988.*

[14] K. Li, Performance Analysis of Power-Aware Task Scheduling Algorithms on Multiprocessor Computers with Dynamic Voltage and Speed, *IEEE Transaction on Parallel and Distributed Systems. Vol. 19, No. 11. November 2008.*

[15] W. Al-Salih, S. Akl and H.S. Hassanein, Cooperative ad hoc computing: Towards Enabling cooperative processing in wireless environments, *International Juornal of parallel, Emergent and distributed Systems,23:1, 59-79, 2008.*

[16] Z. Zong, Mais. Nijim, A. Manzanare, and X. Qin, Energy Efficient Scheduling for Parallel application on Mobile Clusters, *Cluster Computing, Volume 11, 2008, 91-113.*

[17] Dick, R., Rhodes, D. and Wolf, W., 1998, TGFF: task graphs for free, *Proceedings of International Workshop on Hardware/Software Codesign, March.*

[18] X. Lu, H. S. Hassanein and S. Akl, Energy Aware Dynamic Task Allocation in Mobile Ad hoc Networks, *International Conference on Wireless Communications and Mobile Computing, June, 2005.*

[19] C. Perkins, and P. Bhagwat, Highly Dynamic Destination-Sequenced distance-Vector Routing (DSDV) for Mobile Computers, *Computer Communication Review, october 1994, 234-244.*

[20] P. Jacquet, P. Muhlethaler, and T. Clausen, A. Laouiti, AQayyum, and L. Viennot, Optimized Link State Routing Protocol for Ad Hoc networks, proceedings of IEEE INMIC, Pakistan 2001.

[21] B. Bellur, and R. Ogier, A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks, *proceedings of IEEE Infocom, June 1999.*

[22] J. Garcia and M. Sphon, Source Tree Adaptive Routing, *Internet Draft, draft-ietf-manet-star, 1999.*

[23] J. Chlamtac, M. Conti, and J. Liu, "Mobile Ad hoc Networking: Imperatives and Challenges", Ad Hoc Network Journal, Volume 1, No. 1, pp. 13-64, January 2003.

[24] S. Kumar, S. Basavaraju, and C. Puttamadappa. "Ad Hoc Mobile Wireless Networks," Auerbach Publications, 2008.

[25] J, Wu, Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, *Auerbach publications, Taylor & Francis Group, 2006.*

[26] C. Perkins, and E. Royer, Ad Hoc On-Demand Distance Vector Routing, IEEE Workshop on Mobile Computing systems and Application, February 1990, 90-100.

[27] V. Park, and M. Corson, A Highly Adaptive distributed Routing Algorithm for Mobile and wireless Networks, Proceeding of IEEE Infocom, April 1997, 103-112.

[28] S. Pawaskar, and H. Ali, Dynamic Energy Aware Task Scheduling Using Run-Queue Peek, *proceeding of the 25th lasted international multi-conference parallel and distributed computing and networks, February 13-15, 2007, Innsbruck, Austria.*

[29] I. Hong, M. Potkonjak, M. Srivastava, Synthesis techniques for low-power hard real-time systems on variable voltage processors. *In: Proc. IEEE Real-Time System Symp., Dec. 1998.*

[30] J. Lorch, A. Smith, Improving dynamic voltage scaling algorithm with PACE. *In: Proc. ACM SIGMETRICS Conf., Cambridge, MA, June 2001.*

[31] K. Li, R. Kumpf, P. Horton and T. Anderson, A Qauntitative Analysis of Disk Driver Power Management in Portable Computers, *In Proceedings of USENIX conference, pp. 279-292,* Winter 1994.

[32] R. Gonzalez, B. Gordon and M. Horowitz, Supply and threshold voltage scaling for low power CMOS, *IEEE Journal of Solid-State Circuits*, 32, *No 8. pp. 1210-1216,* 1997.

[33] L. Benini, A. Bogliolo, and D. Micheli, A survey of design techniques for system-level dynamic power management. *IEEE Trans. Very Large Scale Integr. Syst. 8(3), 299–316 (2000).*

[34] J. Lorch, and A. Smith, Software strategies for portable computer energy management, *IEEE Personal Commun. 5, 60–73 (1998).*

[35] E. Elnozahy, M. Kistler, R. Rajamony, Energy-efficient server clusters. *In: Proceedings of the Workshop on Power-Aware Computing Systems, February 2002.*

[36] M. Srivastava, A. Chandrakasan, and R. Brodersen, Predictive system shutdown and other architectural techniques for energy efficient programmable computation, *IEEE Trans. VLSI Syst. 4(1), 42–55 (1996).*

[37] F. Douglis, P. Krishnan, and B. Bershad, Adaptive disk spin-down policies for mobile computer, *USENIX Symp. Mobile and Location-Independent Computing, pp. 121–137 (1995).*

[38] P. Basu, N. Khan, and T. D. C. Little, "A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks," in *Proc. IEEE ICDCSW' 01*, Apr. 2001, pp. 413–18.

[39] Y. Chen and A. Liestman, Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks, *in Proc. 3rd ACM Int'l. Symp. Mobile Ad Hoc Net. & Comp., June 2002, pp. 165–72.*

[40] C. Lin and M. Gerla, Adaptive Clustering for Mobile Wireless Networks, *IEEE JSAC*, vol. 15, Sept. 1997, pp. 1265–75.

[41] J. Wu and H. Li, On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks, *Proc. 3rd Int'l. Wksp. Discrete Algorithms and Methods for Mobile Comp. and Commun.*, 1999, pp. 7–14.

[42] Y. CHEN, A. L, and J. LIU, CLUSTERING ALGORITHMS FOR AD HOC WIRELESS NETWORKS, *Ad Hoc and Sensor Networks. Nova Science Publishers, 2004*

[43] B. Das and V. Bharghavan, Routing in Ad Hoc Networks Using Minimum Connected Dominating Sets, in *Proc. IEEE ICC'97*, June 1997, pp. 376–80.

[44] A. MaDonald and T. Znati, A Mobility-based Frame Work for Adaptive Clustering in Wireless Ad Hoc Networks, *IEEE JSAC*, vol. 17, Aug. 1999., pp. 1466–87.

[45] J. Yu and P. Chong, 3hBAC (3-hop between Adjacent Clusterheads): a Novel Non-overlapping Clustering Algorithm for Mobile Ad Hoc Networks, *in Proc. IEEE Pacrim'03, vol. 1, Aug. 2003, pp. 318–21.*

[46] T. J. Kwon *et al.*, Efficient Flooding with Passive Clustering — an Overhead-Free Selective Forward Mechanism for Ad Hoc/Sensor Networks, *in Proc. IEEE, vol. 91, no. 8, Aug. 2003, pp. 1210–20.*

[47] A. McDonald and T. Znati, Design and Performance of a Distributed Dynamic Clustering Algorithm for Ad-Hoc Networks, in *Proc. 34th Annual Simulation Symp.*, Apr. 2001, pp. 27–35.

[48] A. Amis and R. Prakash, Load-Balancing Clusters in Wireless Ad Hoc Networks, in *Proc. 3rd IEEE ASSET'00*, Mar. 2000, pp. 25–32.

[49] J. Ryu, S. Song, and D.-H. Cho, New Clustering Schemes for Energy Conservation in Two-Tiered Mobile Ad-Hoc Networks, in *Proc. IEEE ICC'01*, vol. 3, June 2001, pp. 862–66.

[50] T. Ohta, S. Inoue, and Y. Kakuda, An Adaptive Multihop Clustering Scheme for Highly Mobile Ad Hoc Networks, in *Proc. 6th ISADS'03*, Apr. 2003.

[51] M. K, and B. K, Multi-layer Clusters in Ad-hoc Networks- An Approach to Service Discovery, *Proceedings of the First International Workshop on Peer-to-Peer Computing, 2002.*

[52] M. Chatterjee, S. Das, and D. Turgut, An On-Demand Weighted Clustering Algorithm (WCA) for Ad hoc Networks, in *Proc. IEEE Globecom'00*, 2000, pp. 1697–701.

[53] H. Saputra, M. Kandemir, N. Vijaykrishnan, M. Irwin, J. Hu, CH, Hsu and U. Kremer, Energy-Conscious Compilation Based on Voltage Scaling, *In Proceedings of LCTES02-SCOPES02*, June 2002.

[54] Y. Lu, L. Benini and G. Micheli, Low-Power Task Scheduling for Multiple Devices, *In International Workshop on Hardware/Software Codesign, pp. 39-43, 2000.*

[55] J. Ahmed, and C. Chakrabarti, A Dynamic Task Scheduling Algorithm for Battery Powered DVS Systems, *Proc. ISCAS, 813-816, 2004.*

[56] Alghamdi, M., Xie, T., Qin, X.: PARM: A power-aware message scheduling algorithm for real-time wireless networks. *In Proc. ACM Workshop Wireless Multimedia Networking and Performance Modeling, Montreal, Oct. 2005.*

[57] Hong, Kirovski, D. Qu, G. Potkonjak, M. Srivastava, M, Power optimization of variable voltage core-based systems, *In Proc. Design Automation Conf. (1998).*

[58] J. Flinn, Extending Mobile Computer Battery Life through Energy-Aware Adaptation, *Ph.D. Dissertation. Carnegie Mellon University, Computer Science Department, 2001.*

[59] X. Gu, A. Messer, I. Greenberg, D. Milojicic and K. Nahrstedt, Adaptive Offloading for Pervasive Computing, *In Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2003.

[60] T. Xie, X. Qin, and M. Nijim, Solving energy-latency dilemma: task allocation for parallel applications in heterogeneous embedded systems, *In: Proc. 35th Int'l Conf. Parallel Processing, Columbus, Ohio, Aug. 2006.*

[61] Y. Prasanna, V. K, Energy-balanced task allocation for collaborative processing in wireless sensor networks, *Mobile Netw. Appl. 10(1–2), 115–131 (2005).*

[62] [ P. Chowdhury and C. Chakrabarti, Static Task-Scheduling Algorithms for Battery-Powered DVS Systems, *IEEE Transaction on very large scale integration (VLSI) systems, vol.13,NO.2, Feb 2005.*

[63] K. Li, Performance Analysis of Power-Aware Task Scheduling Algorithms on Multiprocessor Computers with Dynamic Voltage and Speed, *IEEE Transaction on Parallel and Distributed Systems. Vol. 19, No. 11. November 2008.*

[64] H. Zheng, R. Buyya, and S. Bhattacharya, Mobile cluster computing and timeliness issues, *Inform. Int. J. Comput. Inform. 23(1), 1999.*

[65] M. Mohamed, V. Devanathan, and D. Ram, A model for mobile cluster computing: design and evaluation, *In International Conference on Computer Science and its Applications (2003).*

[66] M. Mohamed, A. Srinivas, and D. Janakiram, Moset: an anonymous remote mobile cluster computing paradigm, *J. Parallel Distributed Comput. (JPDC) 65(10), 1212–1222 (2005).*

[67] J. Flinn, D. Narayanan and M. Satyanarayanan, Self-Tuned Remote Execution for Pervasive Computing, *In Procceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001.

[68] A. Olsen, F. Fitzek, and P. Koch, Energy aware computing in Cooperative wireless networks, *In International Conference on Wireless Networks, Communications, and Mobile Computing (WirelessCOM'05), Maui,Hawaii, USA.*

[69] A. Olsen, F. Fitzek, and P. Koch, Evaluation of cooperative task computing for energy aware wireless networks, *In International Workshop on Wireless Ad-hoc Networks (IWWAN'05), London, England.*

[70] A. Olsen, F. Fitzek, and P. Koch, Optimizing the number of cooperating terminals for energy aware task computing in wireless networks, *In Wireless Personal Multimedia Communications (WPMC'05) Symposia, Aalborg, Denmark.*

[71] J. Kurose, K. Ross, "Computer Networking," Pearson Education, 4th ed., 2008.

# VITA

**Personal Information**

Surname, Name: Bokar, Ali

Nationality: Yemen (YE)

Date and Place of Birth: 03-06-1970

Marital Status: Maried

email: bookerali@yahoo.com

**Education:**

**2003- Present :** PhD. in Computer Engineering Middle East Technical University (METU), Turkey

**1997-2001 :** Ms. in Computer Science AL al-Bayt University, Jordan

(Ranked 1st)

***Thesis Title**: The Implementation of Distributed Parallel Linda (ParLin) on the PowerXplorere Machine*

**1992- 1996 :** Bs. in Computer Science Mosul University, Iraq (Ranked 13th from 60 students)

**Professional Experience:**

**2007- 2008** : Teaching Assistant, Computer Engineering Department METU, Turkey

**2006- 2007** : Computer lab coordinator, Electrical Engineering Department METU, Turkey

**2005- 2006** : Computer maintenance, Mathematics Department METU, Turkey

**1996- 1997** : Instructor, Dar Al-Fiker Dar Al-Fiker Institute, Yemen

**1996- 1997** : Teaching Assistant, Computer Science Department Hadhramout University, Yemen

**Publications:**

**A. Bokar**, M. Bozyigit, and C. Sener "Energy-Aware Dynamic Server Selection and Task Allocation", the 23rd International Symposium on Computer and Information Science ( ISCIS 2008), Istanbul, Turkey, October 2008.

**A. Bokar**, M. Bozyigit, and C. Sener "Scalable Energy-Aware Dynamic Task Allocation", to appear in The Fifth IEEE International Workshop on Heterogeneous Networks ( AINA 2009), Bradford, UK. May 2009.