P-3278

KCT
LIBRARY
COIMBATORE 641 006

RENDEZVOUS SERVICE COORDINATION IN MOBILE AD HOC NETWORKS

By

S.ARUL JOTHI

Reg. No: 0820108003

of

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Coimbatore)

COIMBATORE – 641 006

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements
for the award of the degree
of

MASTER OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

MAY 2010

BONAFIDE CERTIFICATE

Certified that this project report titled "**RENDEZVOUS SERVICE COORDINATION IN MOBILE AD HOC NETWORKS**" is the bonafide work of **Ms.S.ARUL JOTHI (0820108003)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report of dissertation on the basis of which a degree or ward was conferred on an earlier occasion on this or any other candidate.

[illegible] 03/05/10

GUIDE

(Mrs. J.CYNTHIA M.E.,)

[illegible]

HEAD OF THE DEPARTMENT

(Dr.S.THANGASAMY Ph.D.,)

The candidate with **University Register No. 0820108003** was examined by us in Project Viva-Voce examination held on 17/5/10

[illegible] 17/5/10

INTERNAL EXAMINER

[illegible]

EXTERNAL EXAMINER

BANNARI AMMAN
INSTITUTE OF TECHNOLOGY
Stay Ahead

BANNARI AMMAN INSTITUTE OF TECHNOLOGY

TÜV NORD
ISO 9001

(An Autonomous Institution Affiliated to
Anna University - Coimbatore, Approved by AICTE,
Accredited by NBA and NACC with A Grade)
Sathyamangalam - 638 401 Erode District Tamil Nadu India

DEPARTMENT OF COMPUTER APPLICATIONS

2nd NATIONAL CONFERENCE
on
ISSUES AND TRENDS IN ADVANCED COMPUTING (NITAC-2010)

19 - 20 March, 2010

CERTIFICATE

This is to certify that ~~Dr.~~ / ~~Prof.~~ / ~~Mr.~~ / Ms. S. ABUL JOTHI

of KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

has participated & presented a paper titled RENDEZVOUS BASED SERVICE COORDINATION IN MOBILE ADHOC NETWORKS in the 2nd National Conference on Issues and Trends in Advanced Computing organized by the Department of Computer Applications, Bannari Amman Institute of Technology, Sathyamangalam during March 19 - 20, 2010.

Date : 20.03.2010
Place : Sathyamangalam

[illegible] Organizing Secretary

[illegible] Co-convener

[illegible] Convener

ABSTRACT

Efficient routing and service provisioning in MANET is a big research challenge. In centralized directory-based schemes, some mobile nodes hold the service directory to assist the communications between service providers and clients. Although service co-ordination is easier, such centralized management is hard to scale and the centralized directories lead to bottlenecks. Later Hybrid and Distributed schemes constructed local directories which form the backbone of the network. But its topology-based scheme is still hard to scale to a larger network. In service provision framework, Services are first registered with a core node and is then advertised. The service requests are forwarded by the coordinator to the service provider and the service provider delivers the appropriate service using some multicast algorithms. In the proposed coordination scheme there is one coordinator for each of the zone and one rendezvous regional coordinator that coordinates the services, service requests and service providers. This project deals with an adaptive service coordination scheme, which tracks the location of all nodes and delivers the service using geographic location based routing algorithm. Using an adaptive coordinator election mechanism, only one node per zone is elected as coordinator unlike the other methods found in the literature where one coordinator per group per zone is selected. Though the overhead of a single node is increased, there is overall reduction of encoding, membership maintenance and control overhead. The coordinator changes adaptively according to the resource availability.

ஆய்வுச்சுருக்கம்

தானியங்கி வலைச்சேவைக்கு ஆக்கபூர்வமான வழிமுறையைச் செய்வது என்பது மிகப்பெரிய சவாலான ஆராய்ச்சியாகும். இதில் நடுநிலைப் படுத்தப்பட்ட கையேட்டுப்பிரிவில் சில தானியங்கி சந்திப்பு வாடிக்கையாளருக்கும், ஒருங்கினைப்பாளருக்கும் இடையே தொடர்பேற்படுத்த வழிவகை செய்கிறது. இருந்தாலும் சேவை ஒருங்கினைப்பானது எளிதாக உள்ளது. மையப்படுத்தபட்ட நிர்வாக மேலாண்மையும், மையப்படுத்தபட்ட கையேட்டுப்பிரதிப்படுத்தப் பிரித்து வகைபடுத்தப்பட்ட திட்டங்கள் உள்ளமைவு கையேட்டுப் பிரதிகள் தயாரிப்பதற்கும், வலைச்சேவைகும் முதுகெலும்பாக உள்ளது. பரந்தப்பட்டதும் விரிவானதுமான வலைச்சேவைக்கு இவை பொருந்தாது. சேவை கட்டமைப்பு வசதிக்கு சேவை ஒலிபதிவானது எலிதாகிறது. சேவைக்கான கோரிக்கைகள் முதலில் ஒருங்கினைபாளர்களுக்கு அனுப்பப்படுகிறது.

பிறகு அது பன்முனைப் படுத்தப்பட்டு திட்டங்களாக ஒருமுகப்படுத்தப்படுகிறது. இந்த ஆய்வின் மூலமாக முன்மொழியப்பட்ட ஒருங்கினைப்பானது மண்டல ஒருங்கினைப்பாளர் முதன்மை ஒருங்கினைப்பாளர் வழியாகத்தகவல் பரிமாற்றம் நடைபெருகிறது. இவ்வாய்வுத்திட்டத்தின் மூலமாக தேவைக்கேற்ப சேவைகளை ஏற்றுக்கொள்வதற்க்கும், இடத்திற்கு தகுந்தவாறு சேவைகளை பிரித்தாளுகை செய்வதற்க்கும் வழிவகை செய்கிறது. கள ஆய்வில் கண்டறியப்பட்டது போல் அல்லாமல் ஒரு மண்டலத்திற்கு ஒரு ஒருங்கினைப்பளர் மட்டும் தேர்வு செய்ய வழிவகை செய்கிறது. அதனுடைய முதன்மையான செயல்பாடு அதிகரித்தாலும் உறுப்பினர் பராமரிப்பும் தகவல் பரிமாற்றமும் குறைகிறது. தேவைக்கேற்ப மாற்றிக்கொள்ளும் வழிவகையை ஒருங்கினைப்பாளர்க்குத் தருகிறது.

ACKNOWLEDGEMENT

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E.**, for giving this opportunity to pursue this course.

I would like to thank to **Dr.S.Ramachandran, Ph.D.**, *Principal* for providing the necessary facilities to complete my thesis.

I take this opportunity to thank **Dr.S.Thangasamy Ph.D.**, *Dean, Head of the Department,* Computer Science and Engineering, for his precious suggestions.

I thank all project committee members for their comments and advice during the reviews. Special thanks to **Mrs.V.Vanitha M.E., (Ph.D),** *Assistant Professor*, Department of Computer science and Engineering, for arranging the brain storming project review sessions.

I register my hearty appreciation to the Guide **Mrs.J.Cynthia M.E.**, *Assisstant Professor,* Department of Information Technology, my thesis advisor. I thank for her support encouragement and ideas. I thank her for the countless hours she has spent with me, discussing everything from research to academic choices.

I would like to convey my honest thanks to all **Teaching** staff members and **Non Teaching** staffs of the department for their support. I would like to thank all my classmates who gave me a proper light moments and study breaks apart from extending some technical support whenever I needed them most.

I dedicate this project work to my **parents** for no reasons but feeling from bottom of my heart, without their love this work wouldn't possible.

TABLE OF CONTENTS

LIST IF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MANET	: Mobile Ad-hoc Network
QoS	: Quality of service
MAC	: Medium Access Control
TCP	: Transmission Control Protocol
GPS	: Global Positioning System
HRPM	: Hierarchical Rendezvous Point Multicast Protocol
RP	: Rendezvous Point
GID	: Group Identifier
AP	: Access Point
ACK	: Acknowledgement
SR	: Service Requestor
ZID	: Zone Identifier
SP ID	: Service Provider Identifier
EF	: Eligibility Factor
NS	: Network Simulator

CHAPTER 1

INTRODUCTION

A mobile ad hoc network (MANET), sometimes called a mobile mesh network, is a self-configuring network of mobile devices connected by wireless links. Each device in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently. Each must forward traffic unrelated to its own use, and therefore be a router. The primary challenge in building a MANET is equipping each device to continuously maintain the information required to properly route traffic.

The growth of laptops and 802.11/Wi-Fi wireless networking have made MANETs a popular research topic since the mid- to late 1990s. Many literature papers evaluate protocols and their abilities, assuming varying degrees of mobility within a bounded space, usually with all nodes within a few hops of each other and usually with nodes sending data at a constant rate. Routing protocols are then evaluated based on the packet drop rate, the overhead introduced.

Multicast is a fundamental service for supporting collaborative applications among a group of mobile users. Unlike in the wired Internet, multicast in MANETs is faced with a more challenging environment. In particular, multicast in MANETs needs to deal with node mobility and, thus, frequent topology changes, a variable quality wireless channel, constrained bandwidth, and low memory and storage capabilities of nodes. Additionally, unlike in the wired Internet, nodes in a MANET can be modified at the network layer to provide group communication support. This reduces the need for overlay-based group communication that has been popular in the Internet.

Both unicast and broadcast traffic are easy for networks to implement; data packets will either be delivered to a single unique destination, or they will be propagated throughout the network for all end users. Supporting multicast traffic is considerably more complex because participants must be identified, and traffic must be sent to their specific locations.

The network should also refrain from sending traffic to unnecessary destinations to avoid wasting valuable bandwidth. Service providers are concerned about the effects of traffic on their networks. Service providers do not support broadcast traffic (no traffic type needs to be delivered to *all* Internet users). However, multicast delivery is in increasing demand. Applications such as data casting (news, stock tickers, etc.), video and audio transmissions, and training seminars (also called *webinars*) all depend on multicast technology. These applications are designed to deliver identical packets to a large number of receivers. The packets must be replicated at an exponential rate – the resulting bandwidth requirements and routing overhead associated with these applications can be quite daunting.

Delivering multicast packets over a large network is a complex process. There are several components of this process that must take place to successfully establish multicast communication. The first step is the identification of the receivers. All the hosts that want to receive a particular stream of multicast information must identify themselves to the network. This is called the registration process, and it is facilitated with a unique set of IP addresses (called Class D addresses) that are reserved specifically for multicast communications. The receivers register with a particular group (for example, to attend an individual webinar) – the concept of "groups" is central to multicast data delivery.

Once receivers join their respective groups (it is likely that a single receiver will join several groups), the network must deliver the multicast traffic to the correct end stations. On the Internet, a routing protocol must be used to determine the appropriate forwarding paths to all of the registered receivers; In addition, the transmissions from the data source must be replicated at some point, so that the information can be received in multiple locations simultaneously. The delivery process is facilitated by a multicast routing protocol. There are several multicast protocols that the user can choose from – each one has its own unique strengths and weaknesses.

Most multicast data transmissions are uni-directional. Typically, a single host will transmit information (such as stock ticker updates or the content for a seminar) that will be received by multiple end stations. Although there are some new IETF proposals for bi-directional multicast traffic, the focus is on one-way, point-to-multipoint transmissions.

Multicast data streams do not support reliable upper-layer protocols such as TCP. Since a transmitter or source does not know how many downstream workstations are receiving the data, it is impossible to maintain reliable TCP connections with all of the end users. Instead, the best-effort based User Datagram Protocol (UDP) is commonly used for multicast traffic. Whenever a packet (such as a stock ticker update) is missed by a receiver, it is simply lost and not retransmitted.

Service provisioning:

Service related protocols in MANET are mainly focused on service discovery and this discovery schemes are normally integrated with different kinds of routing protocols namely reactive, proactive, hybrid, unicast, multicast and anycast protocols.

Service provisioning is closely related to routing. However, the current service provisioning protocols are normally built on or simply extended from the existing topology-based MANET routing protocols. Consequently, they inherit their limitations (e.g., limited scalability, relatively high control overhead and unreliability.

In MANET routing, there is a trend to develop position-based routing schemes which are more robust and efficient than the traditional topology-based routing schemes. In this work, we make use of the position information in our service provisioning mechanism to increase its scalability and efficiency. Instead of using complicated schemes to manage network topologies, only the positions of service nodes need to be tracked.

A service provision framework will support the following functions:

- **Service Discovery:** Locating the services based on user's requests.
- **Service Delivery:** Delivering relevant service data and control messages. Unicast routing is needed for the delivery between peer devices, while multicast routing would be required to support efficient group communications.

  Geographic Routing :Geographic unicast routing assume mobile nodes are aware of their own positions through certain positioning system (e.g., GPS), and a source can obtain the destinations's position through some location service.

  An intermediate node makes packet forwarding decisions based on its knowledge of the neighbors' positions and the destination's position inserted in the packet header by source. As the forwarding decisions are only based on the local topology, geographic routing is more scalable and robust in a dynamic environment. However, these protocols adopt a proactive beaconing scheme to maintain the local topology no matter if there are data traffics.

To reduce control overhead under light traffic, in neighbors of the forwarding node contend for packet forwarding according to their distances to the destination. These simple contention-based schemes may result in redundant forwarding. And lead to large collision probability under high traffic load.

- **Service coordination:** A service request may need to be satisfied by several candidate providers. To enable coordination, the service provision framework should be able to track a group of service providers and their services. Hence efficient and scalable service and membership management is required to facilitate the selection of appropriate providers and the collaboration among multiple providers.

  Existing schemes for service coordination are,

Centralized and Distributed Directory: References developed a distributed service discovery architecture, which relies on a topology. It consists of two independent components:

- Formation of a virtual backbone and
- Distribution of service registrations, requests, and replies.

The first component creates a mesh structure from a subset of a given network graph that includes the nodes acting as service brokers and a subset of paths (which we refer as virtual links) connecting them. Service broker nodes constitute a dominating set, i.e. all the nodes in the network are either in this set or only one-hop away from at least one member of the set.

The second component establishes sub-trees rooted at service requesting nodes and registering servers for efficient dissemination of the service discovery probing messages.

Simulation results were provided for comparison of performance measures, i.e. latency, success rate, and control message overhead, when different architectures and network support mechanisms are utilized in service discovery.

CHAPTER 2

LITERATURE SURVEY

2.1.Challenges Faced in MANET [9]:

Regardless of the attractive applications, the features of Manet introduce several challenges that must be studied carefully before a wide commercial deployment can be expected. These include:

- **Routing:** Since the topology of the network is constantly changing, the issue of routing packets between any pair of nodes becomes a challenging task. Most protocols should be based on reactive routing instead of proactive.
- **Security and Reliability:** An ad hoc network has its particular security problems due to nasty neighbor relaying packets. Further, wireless link characteristics introduce also reliability problems, because of the limited wireless transmission range, the broadcast nature of the wireless medium (e.g. hidden terminal problem), mobility-induced packet losses, and data transmission errors.
- **Quality of service (QoS):** Providing different quality of service levels in a constantly changing environment will be a challenge.
- **Internetworking:** The coexistence of routing protocols, for the sake of internetworking a MANET with a fixed network, in a mobile device is a challenge for the mobility management.
- **Power Consumption:** For most of the lightweight mobile terminals, the communication-related functions should be optimized for less power consumption.

MANET is a highly dynamic environment, so the traditional well established multicasting protocols cannot be deployed directly to it. Some modification and extension should be made while considering all the constraints, such as dynamic network topology, limited bandwidth and power. The new protocols should avoid global flooding, should dynamically build the routes, and should update both routes and memberships periodically.

2.2. Service Provisioning in Scalable Geographic Service Provision Framework for Mobile ad Hoc Networks [1]:

In our framework, the sending of control and data information in the service delivery can be implemented through our unicast and multicast protocols. The other two functions required in service provision can be supported by our hierarchical service and resource management architecture as discussed in this subsection.

In centralized directory-based schemes, some mobile nodes hold the service directory to assist the communications between service providers and clients. Although service co-ordination is easier, such centralized management is hard to scale and the centralized directories lead to bottlenecks. In, local directories are constructed and form the backbone of the network. A request will be searched in a local directory or multiple directories by distributed searching. Its topology-based scheme is still hard to scale to a larger network (e.g., with several hundreds of nodes) and lead to bottle neck and Scalability problems.

2.2.1. Virtual hierarchy:

The hierarchical structure is designed to be flexible. Specifically, the number of management layers in the structure is adjusted according to the capability of the coordinators, the density of service nodes and the service requirements. We define *service coordinator* at each layer to facilitate service management. According to the management range and layer, a coordinator can be classified as local, regional coordinator. The functions supported by a coordinator are also kind of services provided to other wireless nodes. An example of the hierarchical structure is shown in Fig. 2.2.1. The whole network is divided into size-manageable square zones. The zone-structure is virtual and constructed and maintained only on management need.

The RC only needs to track the aggregated information of those zones having SNs instead of every SN. Although a two-tier structure is enough for a normal MANET, in order to scale to a larger network or manage a larger number of SNs, more layers can be introduced between RC and LCs.

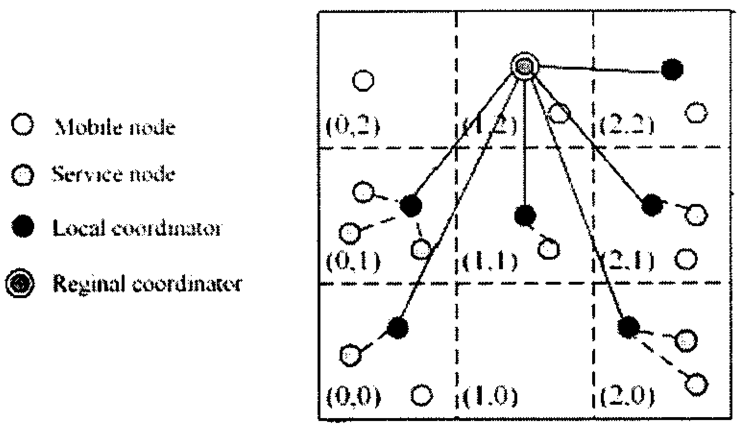


Fig. 2.1. Virtual Hierarchy

2.2.2. Service discovery:

The service discovery follows our hierarchical structure from the bottom layer to the top layer, i.e., first searching for the services and resources locally, then regionally, and at last globally. As our service domains are formed based on geographic information, this searching procedure naturally follows *proximity principle*. Specially, when a service requestor (SR) wants to request one or multiple services, it will send a Query message with service descriptions (including service IDs and parameters) to an appropriate service coordinator. When there is a LC in its zone, it issues the Query to its LC; otherwise, if the SR knows RC, it will issue Query to RC. When neither LC nor RC is known, SR will send Query to GC directly or through GCZone. If having no information on GC, GCZone, RC or LC, SR can start an expanded ring search, which is actually a fully distributed searching as in some service discovery protocols.

When a LC receives a Query or it itself has service requests, if any of the requested services can be satisfied by some local SPs according to its record, the Query will be forwarded to one or more candidate SPs. On receiving the Query, if it can provide the service, the SP will send back a Hit message describing what it can provide to the LC or initial SR (which one to respond to is based on the policy and the service request). If the requested services could not be satisfied locally, the LC will follow the same searching procedure as above and resort to its upper layer service coordinators, and the last option is to look for SPs through expanded ring search.

When a RC or GC receives a Query, it will process the message similarly. If any of the requested services can be satisfied by the service zones recorded, the Query will be further forwarded to the selected candidate zones. Selection criteria such as QoS requirements, geographic closeness will be used when multiple candidates are available. When reaching a destined service zone, the Query will be forwarded to its LC, which will forward the Query to appropriate SPs to check if the services can be supported. Without receiving any Hit message, the initial SR or corresponding coordinator can retry by reissuing the Query or resorting to another SP candidate or upper coordinator.

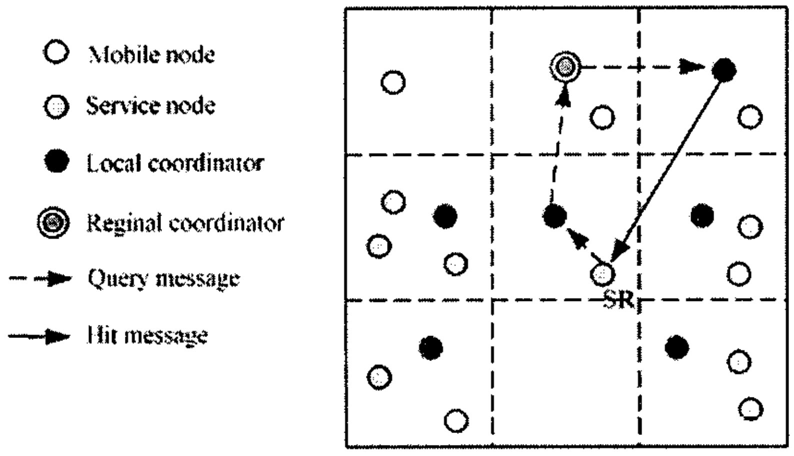


Fig. 2.2. **Service Discovery in Regional Coordinator Layer**

We also use service cache to optimize the service discovery. Each node keeps a service cache. Whenever a node receives a Hit or REGISTER message from a SP, it will cache the SP's position and service information. The information will be removed after caching for a period *Intvalcache*.

During the service discovery, when a non-coordinator node receives a Query, it will look up its service cache for the requested services, and forward the Query to qualified SPs if available, instead of forwarding the Query to destined coordinator.

For a coordinator, if the requested services can not be satisfied by its service records, it will try to search for nonlocal SPs in its service cache, and if this also fails, it will forward the Query to its upper coordinator. Other caching techniques ([11]) can also be applied to further improve the data access performance.

2.2.3. Service Coordination:

A service request may need to be satisfied by several SPs. Our membership and service management framework can efficiently support the search of multiple SPs. During service discovery, when a request can not be fully satisfied by the resources tracked by a LC, it will be further issued to the RC and then GC. Based on the knowledge of larger-range resource and service information, RC and GC can allocate necessary remaining resources to the SR.

2.2.4. Service Delivery:

When the required service is found, the coordinator sends a bit vector (ie, a hit message specifying the position of service). Then the service requestor identifies a path from its source to contact the required service.

2.3. Service Provisioning in Efficient Geographic Multicast Protocol for Mobile Ad Hoc Networks [4]:

EGMP can scale to large group size and network size and can efficiently implement multicast delivery and group membership management. EGMP uses a hierarchical structure to achieve scalability. The network terrain is divided into geographical non-overlapping square zones, and a leader is elected in each zone to take charge of the local group membership management. A zone-based bidirectional multicast tree is built in the network range to connect those zones having group members, and such treestructure can utilize the network resource efficiently.

2.3.1. Service Discovery:

EGMP uses a two-tier structure. The whole network is divided into square zones. In each zone, a leader is elected and serves as a representative of its local zone on the upper tier. The leader collects the local zone's group membership information and represents its associated zone to join or leave the multicast sessions as required.

As a result, a network-range core-zone-based multicast tree is built on the upper tier to connect the member zones. The source sends the multicast packets directly onto the tree. And then the multicast packets will flow along the multicast tree at the upper tier. When an on-tree zone leader receives the packets, it will send them to the group members in its local zone.

2.3.2. Service Coordination:

When a source (S) has data to send and it is not a zLD, it decides whether it has joined the multicast tree by checking the isAcked flag in its membership table. If it is on the multicast tree, it sends the multicast packets to its zLD. When the zLD of a zone on the multicast tree (tZone) receives multicast packets, it forwards the packets to the upstream zone and all the downstream nodes and zones except the incoming one.

2.3.3. Service Delivery:

When a node N has a multicast packet to be forwarded to a list of destinations ($D1,D2,D3, \ldots$), it decides the next hop towards each destination (For a zone, its center is used) using the geographic forwarding strategy. After deciding the next hops, N inserts the list of next hops and associated destinations in the packet header.

An example list is ($N1 : D1,D3;N2 : D2; \ldots$), where $N1$ is the next hop for the destinations $D1$ and $D3$, and $N2$ is the next hop for $D2$. And then N broadcasts the packet *promiscuously* (for reliability and efficiency). Upon receiving the packet, a neighbor node will keep the packet if it is one of the next hops or destinations, and drop the packet otherwise. If the node is a next hop for other destinations, it will continue forwarding the packet similarly as node N.

2.4. Service Provisioning in Hierarchical Rendezvous Point Multicast [3]:

Hierarchal routing [10] is a well known approach to reducing the protocol states in a large scale network. The per-packet encoding overhead of a stateless location-based multicast protocol grows with the group size as O(G), where G is the multicast group size. So, an increase in G severely limits the 5 usability of such protocols.

The main design goal of HRPM is to limit the per-packet overhead to an application-specified constant (ω), irrespective of the increase in G. The value of ω is a parameter of HRPM and can be adjusted based on the amount of overhead that can be tolerated by an application.

To achieve this, HRPM recursively partitions a large multicast group into manageable sized subgroups in which the tree-encoding overhead satisfies the ω

P-3278

K C T
LIBRARY
COIMBATORE - 641 006

constraint. This partitioning is achieved by geographically dividing the MANET region into smaller and smaller *cells*. Such cells form a hierarchy with the root representing the entire region.

Every cell in the hierarchy has an AP (*Access Point*), and the entire region has an RP (*Rendezvous Point*). All members in a leaf cell of the hierarchy form a subgroup and are managed by that cell's AP. Groups of APs are managed recursively, i.e., by the APs of their parent cells. ω is an application parameter and we discuss how HRPM adjusts the hierarchy to meet this ω constraint.

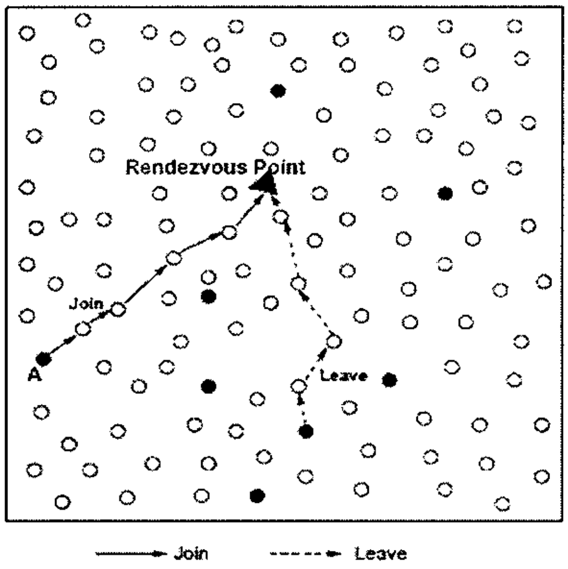


Fig. 2.3. Rendezvous point group management in HRPM

2.4.1. Service Discovery:

Initially when a source needs a service it must be a member of the hierarchy and sends a beacon message to RP to identify the required service location. For this initially a node hashes its group id to get the RP of its zone, which then contacts RP.

2.4.2. Service Coordination:

The source sends an OPEN SESSION message to the RP and receives the membership group vector from the RP. The membership vector is of size d2 bits, with a bit '1' for each cell that contains any group members. This vector is cached by the source. The RP differentially updates (sending only the changes) the source whenever the RP receives a change in membership notification from an AP.

Once the group vector is received, the source can build a virtual overlay tree (the Src ω AP tree) by assuming each active AP as a vertex in a topology graph. The tree is *virtual* since the source does not need to know the actual AP node in each cell; it just needs to hash the GID in the AP's cell.

2.4.3. Service Delivery:

Multicast data packets are delivered down the Src ω AP tree. The per-packet encoding overhead is limited to a constant of d2 bits. Once a data packet reaches an AP, the AP constructs an AP ω Member overlay tree this time using member node identifiers and their actual locations. The AP then encodes the list of destinations and their locations under each branch of the overlay tree in each data packet sent along that branch.

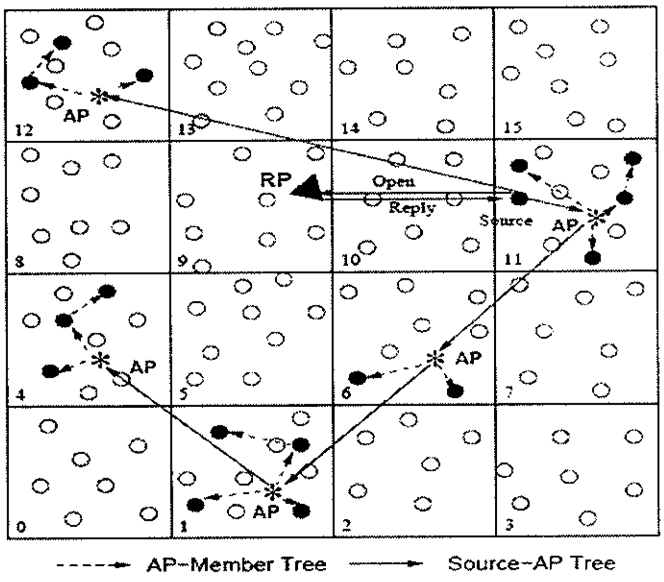


Fig. 2.4.Data Delivery in HRPM

On average, the number of group members in a cell is G d2 where G is the group size. The packet then is delivered to the nodes down the tree, with each node recomputing a tree of the remaining destinations in the list.

Note that the size of this multicast header reduces as the packets travel down the tree and the height of the remaining multicast tree reduces.

2.5. Core Node Election Methods:

A Core Node is a Zone leader that maintains information about its zone members, such as available service, its service providers and location. It gets periodical update from its member node and send its information update periodically to the Regional Leader.

Core nodes are elected based on three algorithms from papers Scalable Geographic Service Provision Framework, Efficient Geographic Multicast Protocol for Mobile Ad Hoc Networks, Hierarchical Rendezvous Point Multicast.

Core node election in SGSP is by considering nodes Zone ID and Node ID. The node with highest Zone ID is elected as Regional Leader and node with highest Node ID is elected as Zone Leader.

Core node election in EGMP is done by the following leader election process, when a node appears in the network, it sends out a beacon announcing its existence. And then it waits for a Intval(min) period for the beacons from other nodes. Every Intval(min) a node will check its neighbor table and decide its zLD under different cases: 1) The neighbor table contains no other zNodes, and it will announce itself as zLD, 2) All the zNodes' flags are unset, that means no zNode has announced the leadership role. If the node is closer to the zone center than other zNodes, it will announce its leadership role through beacon message, 3)More than one zNodes have their flags set, the one with the largest node ID is elected. If the nodes' own flag is set before thechecking, but another node wins as zLD, the node will deliver its multicast table to the zLD. 4) Just one flag is set for its zNodes, the node with flag set is zLD.

Core node election in HRPM is done by using a hash function that takes group ID, number of nodes, and location of each node as input and outputs a location. The node available in that location is considered the leader.

CHAPTER 3

METHODOLOGY

3.1 Core Node Election in Efficient Geographic Multicast Protocol for Mobile Ad Hoc Networks [4]:

In the underneath geographic unicast routing protocols, every node periodically broadcasts a BEACON message to distribute its position. We insert in the BEACON message a flag indicating whether the sender is zLD to ease leader election. Since *rzone* ≤ √*r*2, the broadcasting will cover the hole local zone.

To reduce the beaconing overhead, we enhance the fixed-interval beaconing mechanism in the underneath unicast protocol to a more flexible one. A nonleader node will send a beacon only when its moving distance from last beaconing is larger than or equal to *Dbeacon*, or it moves to a new zone, and the beaconing interval is limited within(*Intvalmin*, *Intvalmax*). A zLD is forced to send out a beacon every period of *Intvalmin* to announce its leadership role.

A zLD is elected through the leader election process. When a node appears in the network, it sends out a beacon announcing its existence. And then it waits for a *Intvalmin* period for the beacons from other nodes.

Every *Intvalmin* a node will check its neighbor table and decide its zLD under different cases:

1) The neighbor table contains no other zNodes, and it will announce itself as zLD.

2) All the zNodes' flags are unset, that means no zNode has announced the leadership role. If the node is closer to the zone center than other zNodes, it will announce its leadership role through beacon message.

3) More than one zNodes have their flags set, the one with the largest node ID is elected. If the node's own flag is set before the checking, but another node wins as zLD, the node will deliver its multicast table to the elected zLD.

4) Just one flag is set for its zNodes, the node with flag set is zLD.

Advantages:

- A zone-based bi-directional multicast tree at the upper tier provides more efficient multicast membership management and data delivery.
- Handles the empty zone problem which is challenging for the zone-based protocols.
- As compared to traditional multicast protocols, this scheme allows the use of position information to reduce the overhead in tree structure maintenance and can adapt to the topology change more quickly.
- Due to the distributed membership management and the distributed tree structure of EGMP, the group members can join the multicast group more quickly than the centralized protocols in which the group members are managed only by the source.

Disadvantages:

- Routing messages preserving energy and network bandwidth is a challenging requirement of paramount importance.
- Packet Forwarding will generally introduce more hops and these extra hops will lead to higher transmission overhead.

3.2 Core Node Election in Hierarchical Rendezvous Point Multicast [3]:

Rendezvous point group management allows multicast group members to leverage geographic hashing for efficient group management. Any node that wants to join a multicast group first hashes the group identifier to obtain the RP's location in the physical domain of the network using a hash function:

$$H(GID) = \{x,y\} \text{ where } x; y \in MANETregion$$

This hashing function takes as input the group identifier (GID) and outputs a location (x- and y-coordinates) contained in the region. Note that we assume that this is a well known hash function that is known by nodes that enter the network through external means or using some resource discovery process.

After obtaining the hashed RP location for the group it wants to join, the node sends a JOIN message addressed to this hashed location. This JOIN message is routed by geographic forwarding to the node that is currently closest to the hashed location in the network. This node is the designated RP at this time. Since there is only one such node at any given time, the JOIN messages from all the group members converge at a single RP in a distributed fashion without global knowledge.

Note that computing the hashed location assumes that all nodes know the approximate geographic boundaries of the network. Such boundary information may be pre-configured at nodes before deployment or discovered using some simple protocol.

To join a hierarchically decomposed multicast group, a node first generates the hashed location for the RP and sends a JOIN message to the RP, same as in the flat domain scenario. After receiving the value of the current decomposition index d of the hierarchy from the RP, the joining node invokes the hashing function with d and its current location to compute the hashed location of the AP of its cell. The node then starts periodically sending LOCATION UPDATE packets to its AP. Such location updates are softstate and serve as a subgroup membership update, i.e., if an AP stops receiving location update from a member, it assumes the member has migrated to another cell.

Upon receiving (or not receiving) a location update from each member, the AP summarizes the membership inside its cell as non-empty (or empty) and further propagates to the RP whenever the membership switches between empty and non-empty. The state the RP needs to keep about the group is just a bit vector of d2 bits with each bit representing whether a member exists in a particular cell or not. Thus the RP can easily encode a large number of APs. For example, 256 APs from 256 cells can be encoded in 32 bytes. Thus for a large multicast group, a two-level HRPM reduces the state required at the RP to d2 bits while requiring the (leaf) AP in each cell to only maintain the addresses and locations of G d2 nodes on average where G is the original size of the multicast group.

The frequency of location update determines the accuracy of the knowledge at the RP/APs and consequently the accuracy of the multicast tree. We use threshold-based updates where each node initiates a LOCATION UPDATE whenever it moves 100m from the location of the last update. This is similar to the strategies used in location services for MANETs. When a node moves into a new cell, it does not immediately send an update to the new AP. Its previous AP can continue to route data using geographic forwarding. When the node moves a certain distance (i.e. 100m) from the location of its last update, it will send a new update to the AP in the new cell.

Note that the group management architecture of HRPM needs to also deal with the situation when nodes of a group are close to each other, i.e. there is locality in the group membership. In such a case, extra overhead is incurred in sending control messages to an RP that may be far away from the cluster of group members. Fortunately, a hierarchy is useful in this scenario as well since a group with locality will send updates primarily to a small set of APs in the clustered cells where the group members are located.

The RP is only sent one update from each AP indicating the existence of members in its cell. Each source only needs to retrieve a bit vector from the RP once to perform data delivery which will be done locally through the nearby APs. Thus, when group membership has geographic locality, HRPM incurs minimal overhead in using an RP. We believe this small overhead is justified given the overall overhead reduction made possible by using a virtual hierarchy.

Advantages:

- HRPM significantly improves the scalability of location-based multicast in terms of the group size.
- HRPM incorporates two key design ideas:

  (1) hierarchical decomposition of multicast groups, and

  (2) use of distributed geographic hashing to construct and maintain such a hierarchy efficiently.
- HRPM constructs and maintains this hierarchy at virtually no cost using distributed hashing; distributed hashing is recursively applied at each subgroup for group management and avoids the potentially high cost associated with maintaining distributed state at mobile nodes.

Disadvantages:

- Keeping track of the RP/AP would require an external location service or some flooding-based mechanism due to mobility in MANETs. This can potentially incur high overhead.

- The group management architecture of HRPM needs to also deal with the situation when nodes of a group are close to each other, i.e. there is locality in the group membership. In such a case, extra overhead is incurred in sending control messages to an RP that may be far away from the cluster of group members.

3.3 Core Node Election in Rendezvous Service Coordination:

In Rendezvous Service Coordination the leaders are elected adaptively considering its remaining battery power, average speed, its frequency of the node assumed the leader till that time (t), and node distance closest to the zone centre. The node with highest Eligibility factor is considered the leader and the leader changes when the Eligibility factor of the existing leader nodes goes down the threshold limit.

$$EF_i(t) = a_1*V_i(t)+a_2*\ I_i(t)+a_3*B_i(t)+a_4(1-E_i(t)).$$

where,

$V_i(t)$ – mobile node average speed at time t.

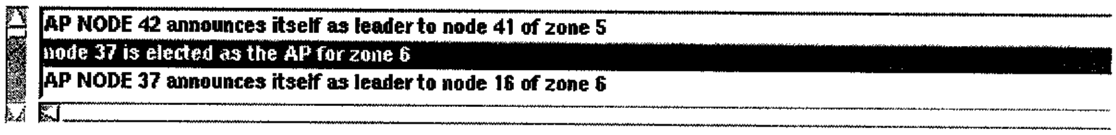
$I_i(t)$ – frequency of the node assumed the leader till the time t.

$B_i(t)$ – remaining battery power in node i at time t.

$E_i(t)$ – node distance closest to the zone center.

a_1 to a_4 – weighting factor. $(0 <= a_i <= 1)$

The node that has highest value of EF will be elected as leader.



```
AP NODE 42 announces itself as leader to node 41 of zone 5
node 37 is elected as the AP for zone 6
AP NODE 37 announces itself as leader to node 16 of zone 6
```

Fig. 3.1. Leader Election

Whenever a mobile node enters a zone it sends a beacon message to its nearest neighbour to get leader's position in the zone. If the new node does not get leader's position within a time interval in that zone, the node considers itself as leader and initiates coordinator election process. The new leader broadcasts its leadership details to all other neighbour nodes in the zone.

Advantages:

- Decreases Handoff by electing Leader adaptively that holds for long time and overhead is reduced by considering only one leader for all the group members in the zone.
- The aggregated Tree structure avoids more confusion, since only one tree is built between the service requestor and service provider of that service closest to requestor.
- Even if the leader node crashes a backup node will be existing maintaining all the information about services, requestors and providers.

Disadvantages:

- More data structures must be maintained by leader node, thus the capability of leader must be considered strictly.
- The leader node must have to save more energy to leads its life and handle coordination.

3.4. Service Registration:

When the Zone Leader is elected, the Service Provider in that Zone Registers its Services with that Zone Leader. Then the Zone Leaders information's are stored, containing the information about Service Providers, location and all services available in that Zone. The following information's are registered,

Table 3.1: Service Provider Table (maintained by Zone leader / Regional leader)

SP ID	SP Loc	ZID	AP Loc	Service ID	Seq No	Lifetime of service

Table 3.2: Service Requestor Table (maintained by Zone Leader)

Member ID	Loc	Group ID's

Table 3.3: Zone Leader Table (maintained by Regional Leader)

Zone Leader	Loc	ZID	EF	(ordered by EF)	Time Stamp

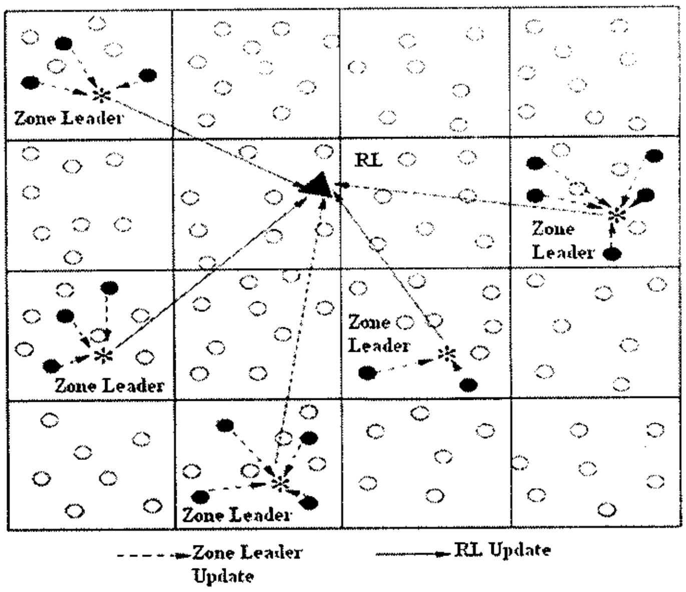
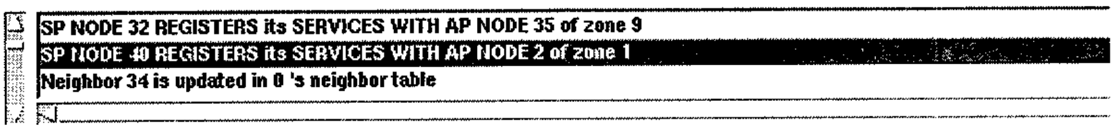


Fig. 3.2. Location Update of all nodes (including leader)

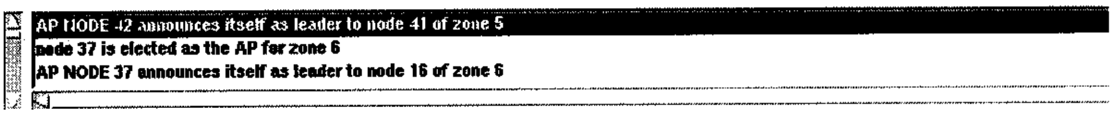


```
SP NODE 32 REGISTERS its SERVICES WITH AP NODE 35 of zone 9
SP NODE 40 REGISTERS its SERVICES WITH AP NODE 2 of zone 1
Neighbor 34 is updated in 0 's neighbor table
```

Fig. 3.3. Service Registration

3.5. Service Advertisement:

The Service Nodes in each Zone advertises its availability and about its service type, its Service provider and its location to the Zone Leader periodically. When the service moves to other Zone it informs its old Zone Leader and advertises its services to the new Zone Leader of its new Zone.



```
AP NODE 42 announces itself as leader to node 41 of zone 5
node 37 is elected as the AP for zone 6
AP NODE 37 announces itself as leader to node 16 of zone 6
```

Fig. 3.4. Service Announcement

3.6. Service Discovery:

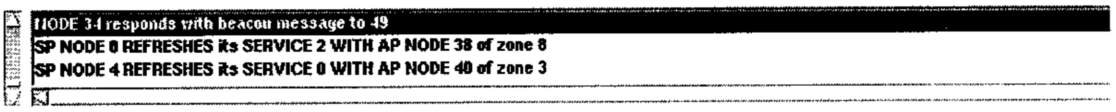
When a Service Requestor has query for service, it identifies its zone location and requests for a Zone Leader location in the zone. After the Zone Leader location is acquired by the requestor, it sends request message to its Zone Leader node. In turn the Zone Leader checks for service availability in its zone and forwards the query to the respective service provider of that service. If the service provider is not available it forwards the query to Regional Leader.

3.7. Rendezvous Service Coordination:

In Rendezvous Based Service Coordination, information about services and the node that provides services are maintained and for efficient tracking and coordination of services, rendezvous node is created so as to reduce overhead and network traffic. Thus the services that can be grouped are identified and coordinated based on the source request.

Initially all the available services are advertised to all Zone Leaders. When a service is needed the Service Requestor contacts Zone Leader by sending a request message, if the service is available the Zone Leader sends a hit message back to the requestor. If the service is not available then the Zone Leader contacts Regional Leader and the Regional Leader confirms the service availability by sending a Hit message to the requestor.

If more than one Service Provider can provide service for the Service Requestor, using (SGSP) [1] the distance between the Zone Leader of Service Requestor and Zone Leader of all the Service Provider is calculated. After this Service Provider with least distance is chosen to provide the required service (i.e., the Service Provider closest to the Service Requestor).



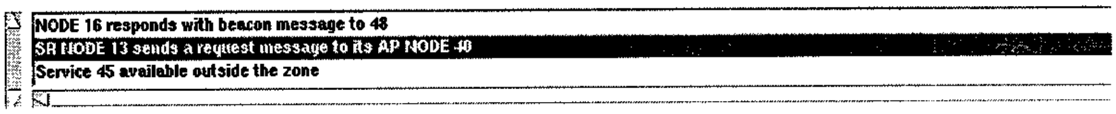
```
NODE 34 responds with beacon message to 49
SP NODE 0 REFRESHES its SERVICE 2 WITH AP NODE 38 of zone 8
SP NODE 4 REFRESHES its SERVICE 0 WITH AP NODE 40 of zone 3
```

Fig. 3.5. Service Coordination

The Service Provider advertises lifetime of service and registers the service details with Regional Leader. Service Provider maintains bit vector of service requestors for each of service. Service Provider periodically delivers service to Service Requestor. Also Service Provider sends service updates to Service Requestor's and Regional Leader.

3.8. Service Delivery:

Only one coordinator is elected in each zone for all groups. The coordinator (i.e. Leader) is changed whenever Eligibility Factor of the node is less than the threshold value, by initiating new election process. If a Service Provider provides 'n' services only one tree is constructed.



```
NODE 16 responds with beacon message to 48
SR NODE 13 sends a request message to its AP NODE 40
Service 45 available outside the zone
```

Fig.3.6. Service Delivery

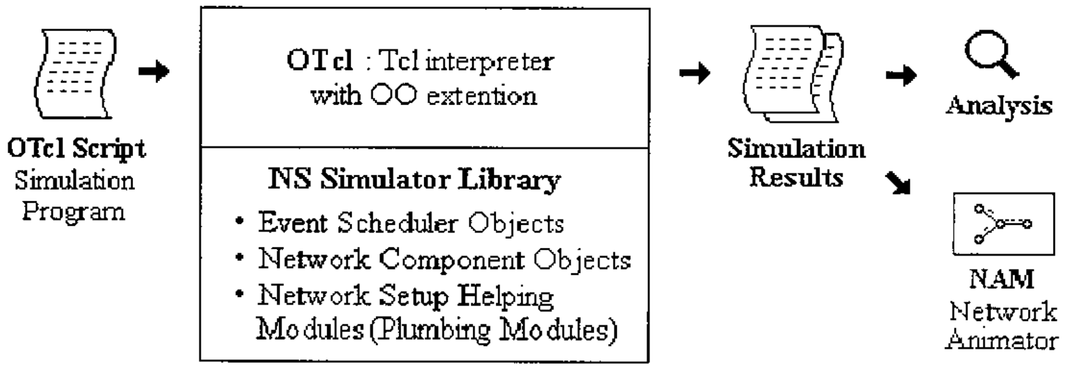
Multicast data packets are delivered down the Source -> Zone Leader tree. Once a data packet reaches a Zone Leader, the Zone Leader constructs a Zone Leader -> Member overlay tree this time using member node identifier and their actual locations. The Zone Leader then encodes the list of destinations and their locations under each branch of the overlay tree in each data packet sent along that branch. The packet then is delivered to the nodes down the tree, with each node recomputing a tree of the remaining destinations in the list.

CHAPTER 4
Implementation

4.1. Simulation Environment:

4.1.1. Network simulator (NS):

Ns-2 stands for Network Simulator Version 2. Ns-2 is a discrete event simulator targeted at network research. Focused on modeling network protocols.


OTcl Script
Simulation Program
OTcl : Tcl interpreter with OO extention
NS Simulator Library
• Event Scheduler Objects
• Network Component Objects
• Network Setup Helping Modules (Plumbing Modules)
Simulation Results
Analysis
NAM
Network Animator

4.1.2. Components of NS

4.1.2.1. Nam, the Network AniMator

- Visualize ns (or other) output
- GUI input simple ns scenarios

4.1.2.2. Pre-processing

- Traffic and topology generators

4.1.2.3. Post-processing

- Simple trace analysis, often in Awk, Perl, or Tcl

4.1.3. Goals of NS

- Support networking research and education
  - Protocol design, traffic studies, etc.
  - Protocol comparison
- Provide a collaborative environment

- Freely distributed, open source
- Share code, protocols, models, etc.
- Allow easy comparison of similar protocols
- Increase confidence in results
  - Models provide useful results in several situations

- It covers multiple layers,
  - Application layer, transport layer, network layer and link layer.
- Supports the simulation of Intserv/diffserv, Multicast, Transport, Applications Wireless(fixed, mobile, satellite)

4.1.4. Two languages:

- C++:
  - Detailed protocol simulations require systems programming language.
  - Byte manipulation, packet processing, algorithm implementations.
  - Run time speed is important.
  - Turn around time(Run simulation, find bug, fix bug, re-compile) is slower.
- **Tcl:**
  - Simulations of slightly varying parameters or configurations.
  - Quickly exploring a number of scenarios.
  - Iteration time(change the model and re-run) is more.

4.1.5. NS-2 Programming Languages

- **NS-2:**

  It is an object oriented simulator, written in C++, with an OTcl(Object Tool Command Language) interpreter as a front-end.
- **Back-end C++:**

  - Defining new agents, protocols and framework.
  - Manipulations at the byte/bit levels.
  - If you have to change the behavior of an existing C++ class n ways that weren't anticipated.

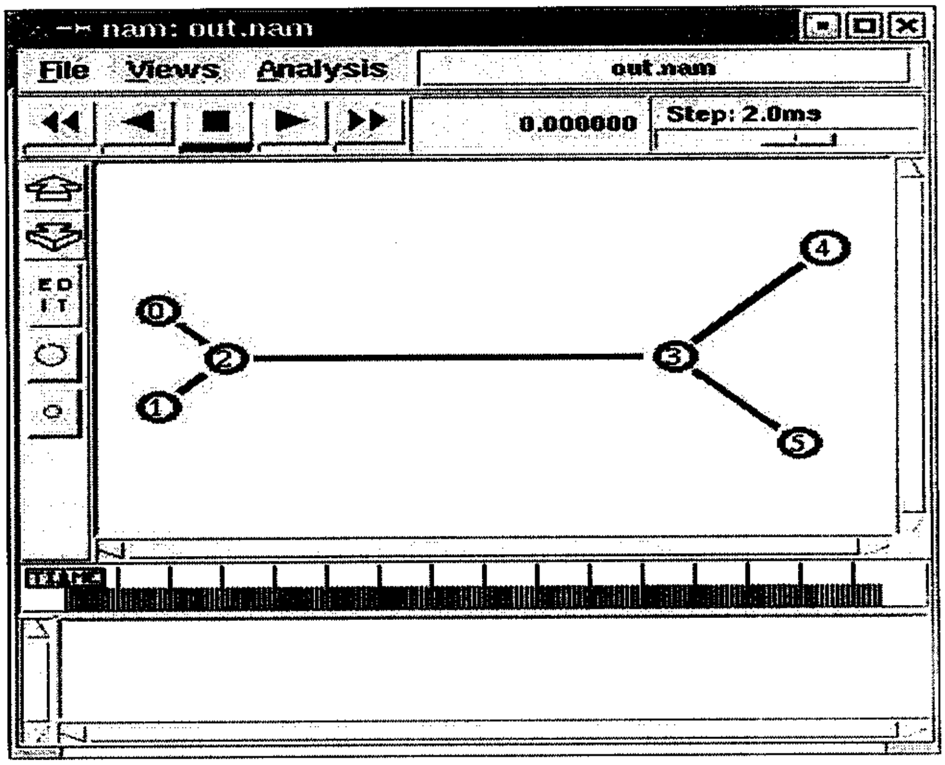
- **Front-end Otcl**
  - Topologies, scenarios, simulations, …
  - Script language (easy topology modifications)
  - If you can do what you want by manipulating existing C++ objects.

4.1.6. Why Two Languages

- Simulator had two distinct requirements
  - Detailed simulation of Protocol (Run-time speed)
  - Varying parameters or configuration (Change model & rerun)
- C++ is fast to run but slower to change
- Otcl runs much slower but can be changed quickly.

4.1.7. Nam (Network AniMator)

"Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet traces."



4.1.8. Gnuplot:

Gnuplot is a command-driven, interactive, function and data plotting program. Gnuplot supports many types of plots in either **2D** and **3D**. It can draw using lines, points, boxes, contours, vector fields, surfaces, and various associated text. It also supports various specialized plot types.
Eg: Gnuplot> plot 'graph.dat'.

4.1.9. Trace File Formats:

Used to trace packets on all links

set tracefd [open simple.tr w]

$ns_trace-all $tracefd

1) **Wired trace file format:**

This trace format is normally used for normal wired operations.

Event	Abbreviation	Type	Value
Normal Event	r: Receive d: Drop e: Error +: Enqueue -: Dequeue	%g %d %d %s %d %s %d %d.%d %d.%d %d %d	
		double	Time
		int	(Link-layer) Source Node
		int	(Link-layer) Destination Node
		string	Packet Name
		int	Packet Size
		string	Flags
		int	Flow ID
		int	(Network-layer) Source Address
		int	Source Port
		int	(Network-layer) Destination Address
		int	Destination Port
		int	Sequence Number
		int	Unique Packet ID

Table. 4.1. Wired Trace File Format

2) Wireless trace file format:

An example of wireless trace output is,

r 40.639943289 _1_ AGT --- 1569 tcp 1032 [a2 1 2 800] ------- [0:0 1:0 32 1] [35 0] 2 0

- The first field is a letter that can have the values r,s,f,D for "received" , "sent" , "forwarded" and dropped. Also M for movement indication.
- The second field is the time.
- The third field is the node number.
- The fourth field is MAC to indicate if the packet concerns a MAC layer, AGT to indicate transport layer, or RTR if it concerns routed packet.
- After the dashes come global sequence number indicator.
- Next field comes for more information on packet type. (tcp, ack, or udp).
- Then comes packet size in bytes.
- Next concerns with MAC layer information. a2 specifies expected time to send data packet, 1 stands for MAC-id, 2 is that for receiving node, and 800 specifies that MAC type is ETHERTYPE_IP.
- The next numbers concern with IP source and destination address.
- The last numbers concerns the tcp information: its sequence number and acknowledgement number.

4.2. Simulation Scenario:

The simulation was performed using the network simulator ns-2. The network field size is 2400mx 2400m, containing 15,20,30,40,50,60 mobile nodes. All the nodes follow the random waypoint mobility model with AODV as the default routing protocol with TCP traffic.

The experimentation is particularly interested in the scalability of multicast routing protocols and service provisioning. The geographic multicasting algorithm is used to deliver the data packets to the group member. The input to this algorithm is service packets with sources and destinations. The output of this algorithm is to deliver the service packet data to the group member.

- **Defining wireless options**

```
# initial configuration
set val(chan)           Channel/WirelessChannel          ;#Channel Type
set val(prop)           Propagation/TwoRayGround;# radio-propagation model
set val(netif)          Phy/WirelessPhy             ;# network interface type
set val(mac)            Mac/802_11                            ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue       ;# interface queue type
set val(ll)             LL                                ;# link layer type
set val(ant)            Antenna/OmniAntenna                ;# antenna model
set val(ifqlen)         15                              ;# max packet in ifq
set val(nn)             15                       ;# number of mobilenodes
set val(rp)             AODV                            ;# routing protocol
set val(x)              2400                   ;# x dimension of topography
set val(y)              2400                   ;# y dimension of topography
set val(seed)           2.0
set val(stop)           100                         ;#time of simulation end
set val(sc)             "output-15-scen"
set val(intvallocal)    4
set val(intvalglobal)   6
set val(energymodel)    EnergyModel
set val(initialenergy)  0.5
```

```
set ns_ [new Simulator]
set tracefd     [open tree.tr w]
$ns_ trace-all $tracefd
set namtrace [open tree.nam w]
$ns_ namtrace-all-wireless
```

- **set up topography object**

```
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
```

- **Create God – General Operations Descriptor**

```
create-god $val(nn)
```

- **Configure node:**

```
$ns_ node-config -adhocRouting $val(rp) \
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
- phyType $val(netif) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace ON \
-movementTrace OFF \
-energyModel $val(energymodel)\
 -rxPower 0.3\
 -txPower 0.6\
 -initialEnergy $val(initialenergy)\
 -batteryModel RTBattery\
 -alpha 35220\
 -beta 0.637\
 -voltage 4.1
```

- **creation of nodes**

```
for { set i 0 } { $i < $val(nn) } { incr i } {
global n
set node_($i) [$ns_ node]
}
```

- **To set the random motion**

```
for { set i 0 } { $i < $val(nn) } { incr i } {
$node_($i) random-motion 0
}
```

- **To set the size of the nodes**

```
for { set i 0} {$i < $val(nn)} {incr i} {
```

```
$ns_ initial_node_pos $node_($i) 60
}
```

- **Loading Scenario File**

```
# loading the initial positions and node movements
puts "Loading scenario pattern..."
source $val(sc)
```

- **Assigning Service Provider and its Services**

```
#assigning every 4th node as a service provider
global sp
set j 0
for {set i 0} {$i < 4} {incr i} {
set sp($i) $j
set j [expr $j+4]
}
$node_(0) set services(0) 1
$node_(0) set services(1) 5
$node_(0) set services(2) 9
$node_(0) set services(3) 13
$node_(0) set services(4) 17
$node_(0) set service_count 5
```

CHAPTER 5

Experimental Results and Discussion

5.1. Performance Evaluation:

The performance analyzed are,

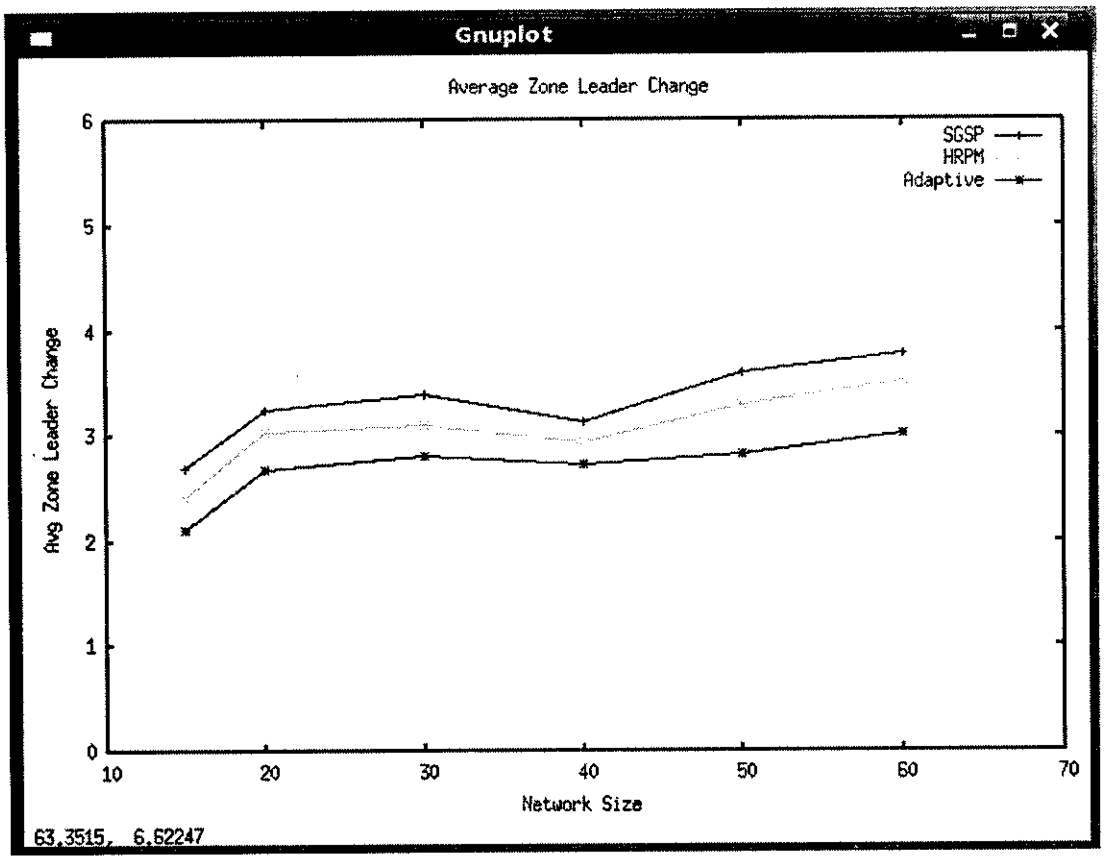
- Packet Delivery Ratio (PDR) – Number of packets delivered to the destination by the number of packets expected to be received.
- Delay – Number of packets sent by Average time received by all receivers.
- Forwarding Cost – Number of packets transmitted by total number of packets received by all members.
- Overhead – Total number of bytes transmitted at MAC layer including ACK in case of unicast transmission.

a) Impact on network size:

- **Network size vs Average zone Leader Change:**

The performances of the three algorithms (SGSP, HRPM, and Adaptive) are evaluated based on the Average Zone Leader Change. The Average Zone Leader Change is calculated by Electing Zone Leader for 900ms with time interval (100ms) by changing the pause time (2,4,5,6,8) of the simulation on an average of 5 simulations.

The graph is plotted by considering network size (15, 20, 30, 40, 50, and 60) on x-axis and Average Zone Leader Change on y-axis.



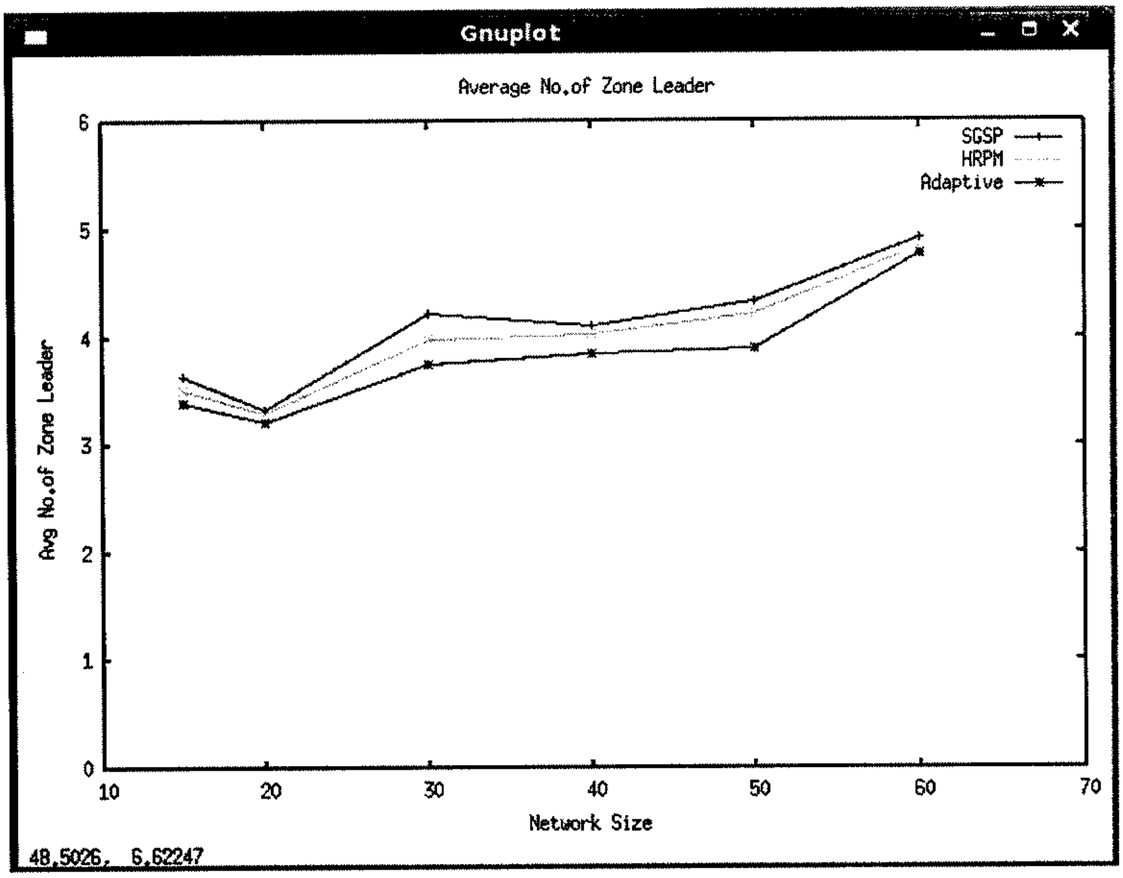
Inference:

From the above graph it is inferred that Zone Leader changes as when the pause time of each simulation is changed, the Adaptive Leader election algorithm is found comparatively efficient than SGSP and HRPM based Leader election.

Because the leaders elected in adaptive leader election algorithm consider their remaining battery power, mobility, and its distance, thus by minimizing the leader change.

- **Network size vs Average no. of Zone leader:**

The graph is plotted by considering network size (15, 20, 30, 40, 50, and 60) on x-axis and Average Zone Leader Change on y-axis. The Average no. of Zone Leader is calculated by Electing Zone Leader for 900ms with time interval (100ms) by changing the seed value (0.0, 1.0, and 2.0) of the simulation on an average of 5 simulations.



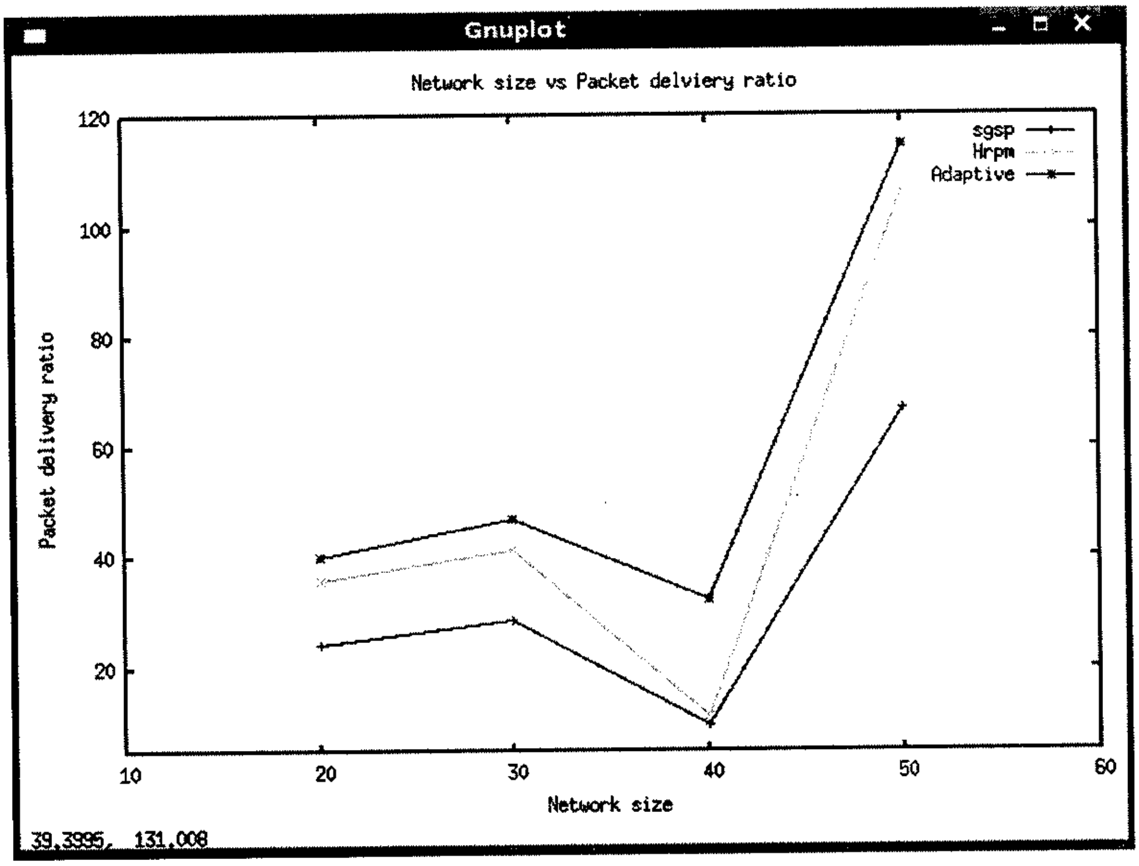
Inference:

From the above graph it is inferred that no. of Zone Leader as when the seed value is changed for each simulation, the Adaptive Leader election algorithm is found comparatively efficient than SGSP and HRPM based Leader election.

Because the leaders elected in adaptive leader election algorithm is based on their remaining battery power, mobility, and its distance, thus by minimizing no. of leaders.

- **Network size vs Packet delivery Ratio:**

  The graph is plotted by considering network size (20, 30, 40, and 50) on x-axis and Packet Delivery Ratio on y-axis. The packet delivery ratio is analyzed by changing seed value (0.0, 1.0, and 2.0).



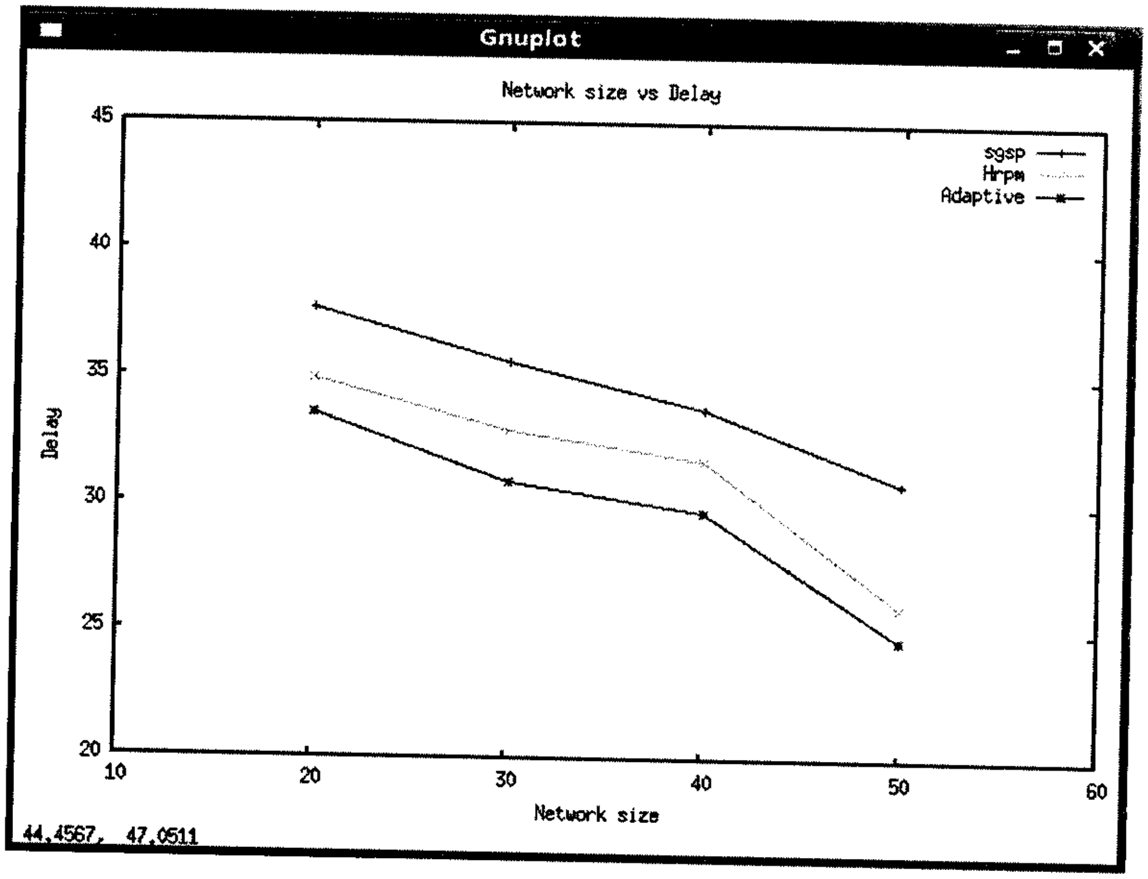
Inference:

From the above graph it is inferred that adaptive algorithm provides about 81.59% of packet delivery ratio which is comparatively efficient than SGSP giving 77.21% packet delivery ratio and HRPM giving 69.64% packet delivery ratio. Also Adaptive algorithm is 19.86% higher than HRPM and 34.92% higher than SGSP.

Thus the adaptive algorithm is efficient because in adaptive coordinator election algorithm the leader nodes are elected based on nodes average speed, distance closest to the centre of zone and remaining energy.

- **Network size vs Delay:**

The graph is plotted by considering network size (20, 30, 40, and 50) on x-axis and Delay on y-axis. The Delay of nodes is analyzed by changing seed value (0.0, 1.0, and 2.0).



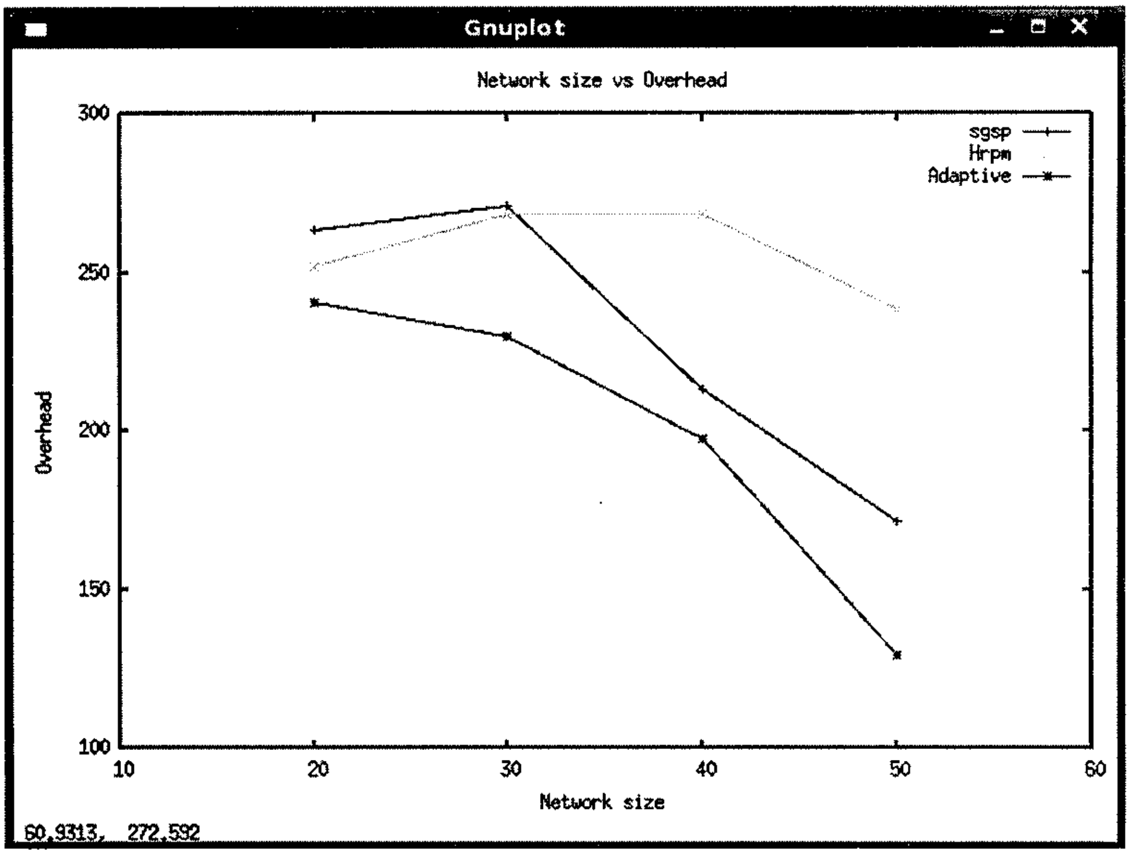
Inference:

From the above graph it is inferred that adaptive algorithm provides about 82.79% of minimized delay which is comparatively efficient than SGSP giving 77.63% minimized delay and HRPM giving 75.42% minimized delay. Also Adaptive algorithm is 5.32% higher than HRPM and 13.83% higher than SGSP.

Thus the adaptive algorithm is efficient because in adaptive coordinator election algorithm the leader nodes are elected based on nodes average speed, distance closest to the centre of zone and remaining energy.

- **Network size vs Overhead:**

  The graph is plotted by considering network size (20, 30, 40, and 50) on x-axis and Overhead on y-axis. The Overhead of nodes is analyzed by changing seed value (0.0, 1.0, and 2.0).



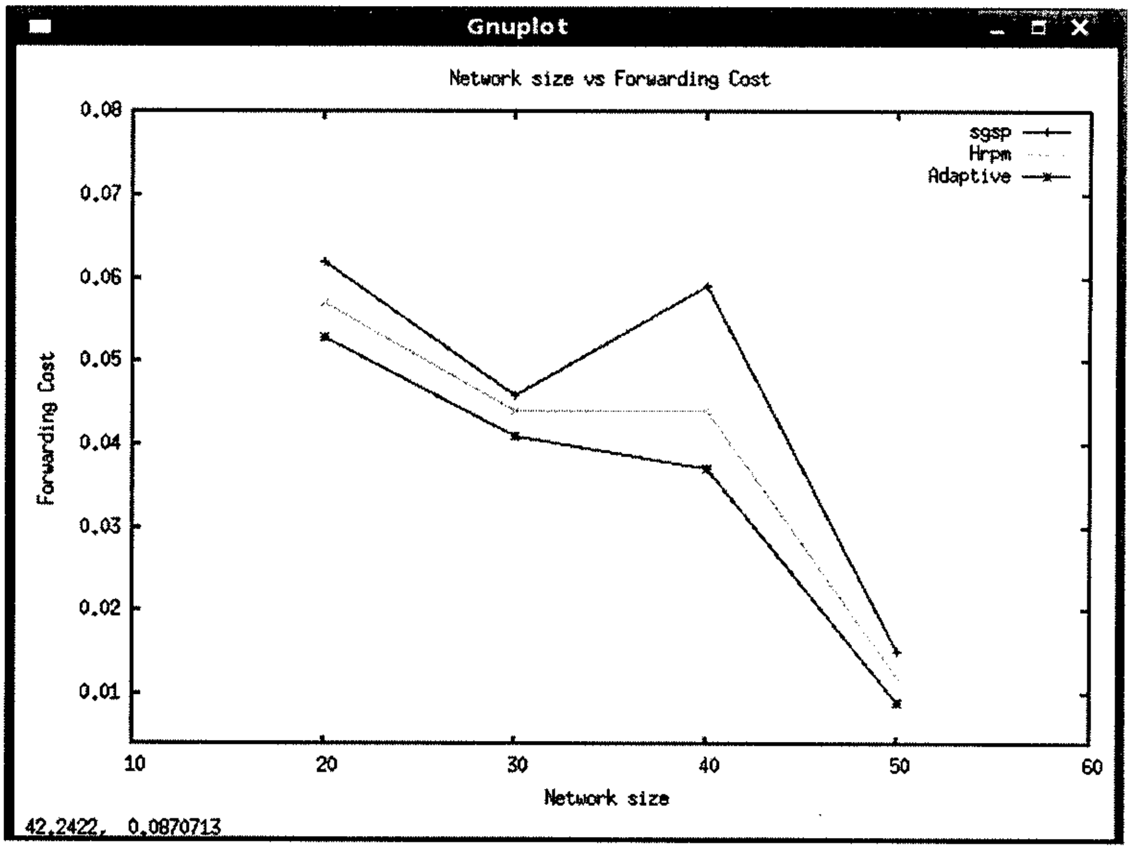
Inference:

From the above graph it is inferred that adaptive algorithm provides about 80.09% of minimized overhead which is comparatively efficient than SGSP giving 73.44% minimized overhead and HRPM giving 78.47% minimized overhead. Also Adaptive algorithm is 22.41% higher than HRPM and 13.28% higher than SGSP.

Thus the adaptive algorithm is efficient because in adaptive coordinator election algorithm the leader nodes are elected based on nodes average speed, distance closest to the centre of zone and remaining energy.

- **Network size vs Forwarding Cost:**

The graph is plotted by considering network size (20, 30, 40, and 50) on x-axis and Forwarding cost on y-axis. The Forwarding cost of nodes is analyzed by changing seed value (0.0, 1.0, and 2.0).



Inference:

From the above graph it is inferred that adaptive algorithm provides about 85.43% of less forwarding cost which is comparatively efficient than SGSP giving 72.79% less forwarding cost and HRPM giving 69.45% less forwarding cost. Also Adaptive algorithm is 12.82% higher than HRPM and 22.44% higher than SGSP.

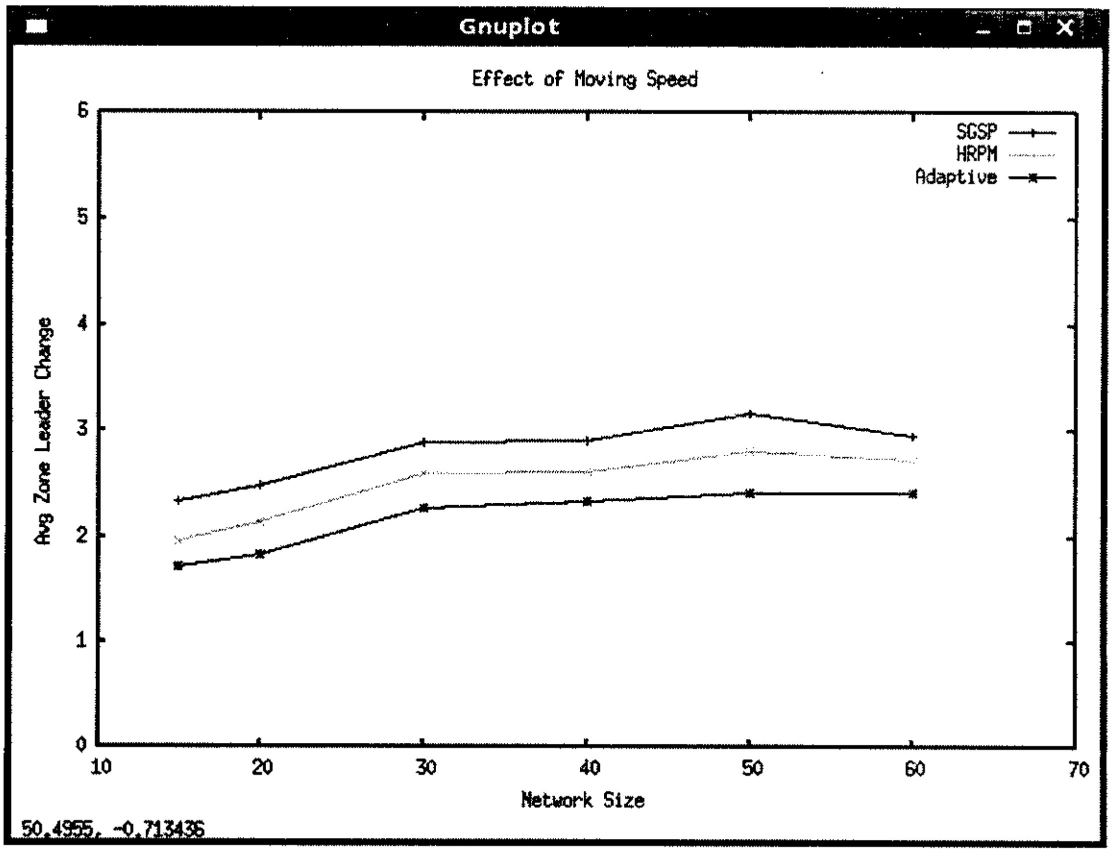
Thus the adaptive algorithm is efficient because in adaptive coordinator election algorithm the leader nodes are elected based on nodes average speed, distance closest to the centre of zone and remaining energy.

b) Impact on Mobility:

- **Network size vs Average Zone Leader change (Effect of Moving Speed) :**

  The graph is plotted by considering network size (15, 20, 30, 40, 50, and 60) on x-axis and Average Zone Leader Change on y-axis.

  The Effect of Moving Speed of a node is analyzed by considering average Zone Leader Change for network size for simulation 900ms with time interval (100ms) by changing the speed (20,40,60,80,100) of the simulation scenario and taking an average of 5 simulations.



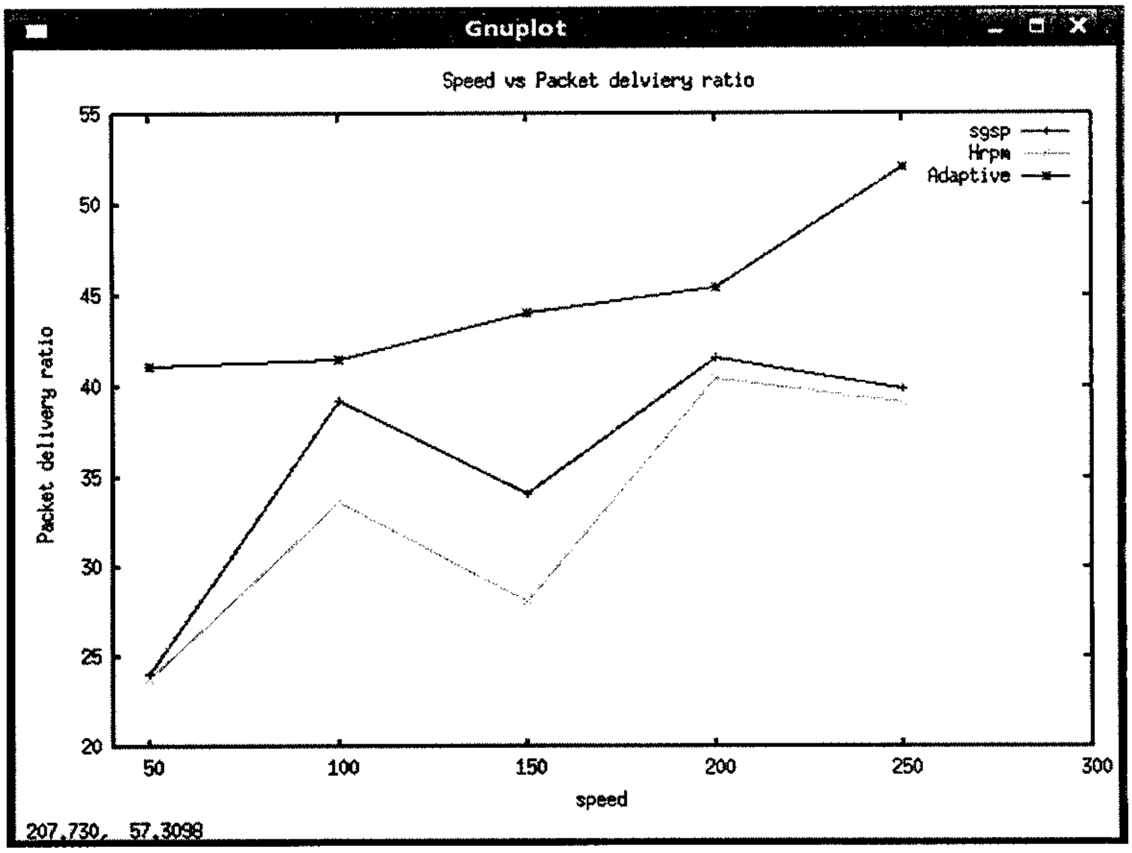
Inference:

From the above graph it is inferred that adaptive algorithm provides about 85.43% of minimized overhead which is comparatively efficient than SGSP giving 72.79% minimized overhead and HRPM giving 69.45% minimized overhead. Also Adaptive algorithm is 12.82% higher than HRPM and 22.44% higher than SGSP.

Because the leaders elected in adaptive leader election algorithm considers the mobility of the leader node and thus the average leader change minimized efficiently.

- **Speed vs Packet Delivery Ratio:**

  The graph is plotted by considering speed (50, 100, 150, 200, and 250) on x-axis and Packet Delivery Ratio on y-axis. The packet delivery ratio is analyzed by changing Pause time (5, 10, 15, 20, and 25).



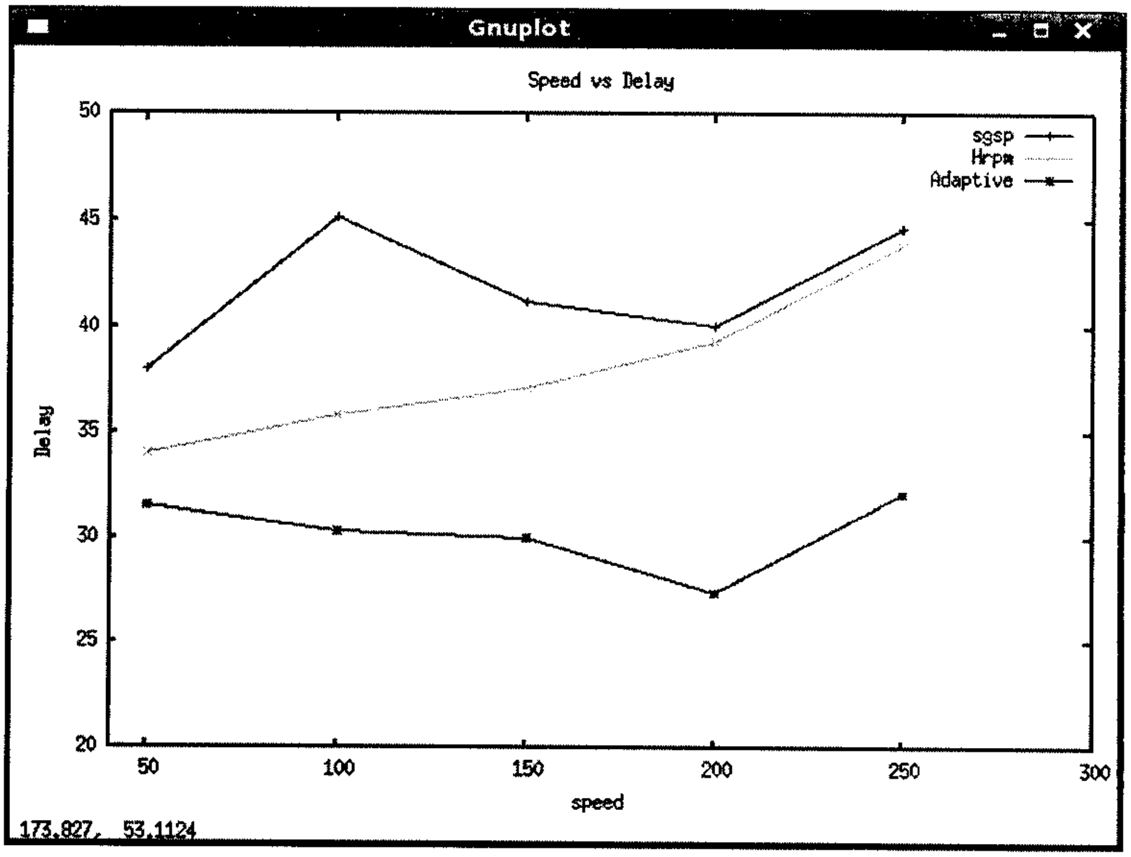
Inference:

From the above graph it is inferred that adaptive algorithm provides about 88.26% of packet delivery ratio which is comparatively efficient than SGSP giving 83.69% packet delivery ratio and HRPM giving 75.53% packet delivery ratio. Also Adaptive algorithm is 35.73% higher than HRPM and 25.29% higher than SGSP.

Thus the adaptive algorithm is efficient because in adaptive coordinator election algorithm the leader nodes are elected based on nodes average speed.

- **Speed vs Delay:**

  The graph is plotted by considering speed (50, 100, 150, 200, and 250) on x-axis and Delay on y-axis. The packet delivery ratio is analyzed by changing Pause time (5, 10, 15, 20, and 25).



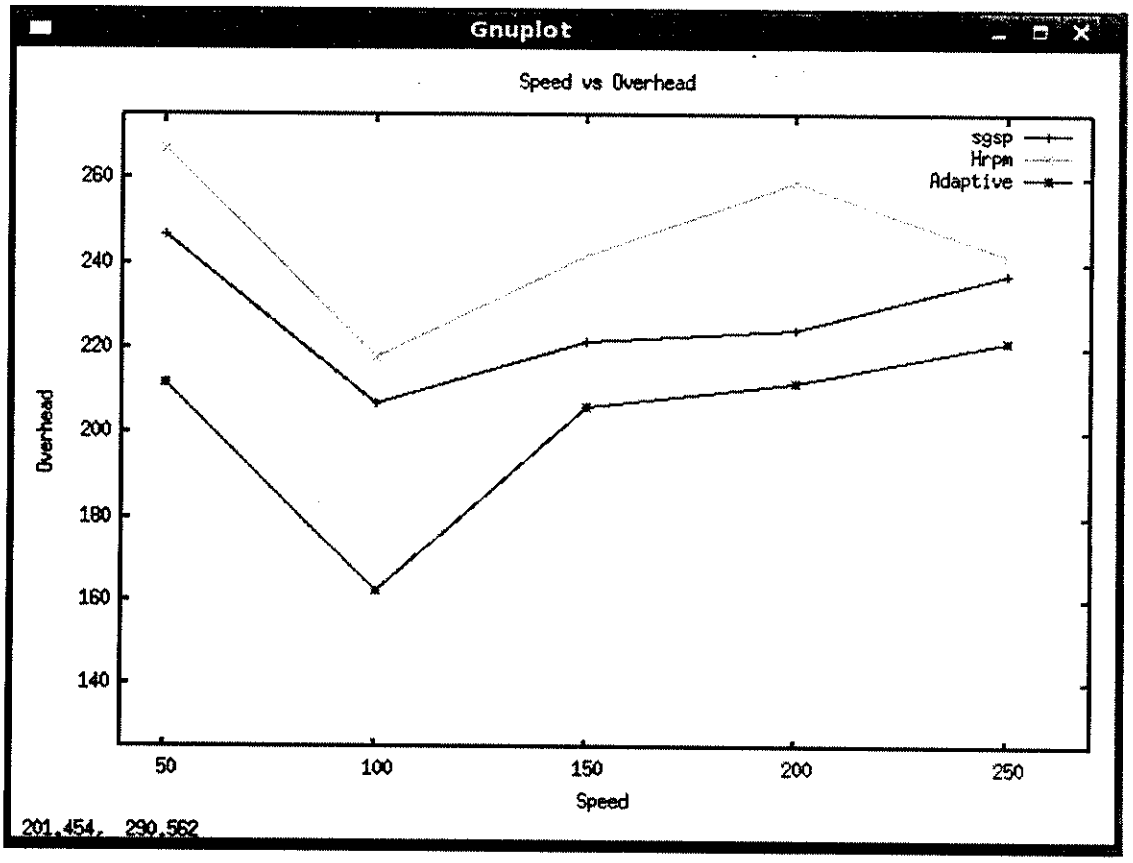
Inference:

From the above graph it is inferred that adaptive algorithm provides about 79.68% of minimized delay which is comparatively efficient than SGSP giving 67.92% minimized delay and HRPM giving 61.56% minimized delay. Also Adaptive algorithm is 20.74% higher than HRPM and 27.62% higher than SGSP.

Thus the adaptive algorithm is efficient because in adaptive coordinator election algorithm the leader nodes are elected based on nodes average speed.

- **Speed vs Overhead:**

  The graph is plotted by considering speed (50, 100, 150, 200, and 250) on x-axis and Overhead on y-axis. The packet delivery ratio is analyzed by changing Pause time (5, 10, 15, 20, and 25).



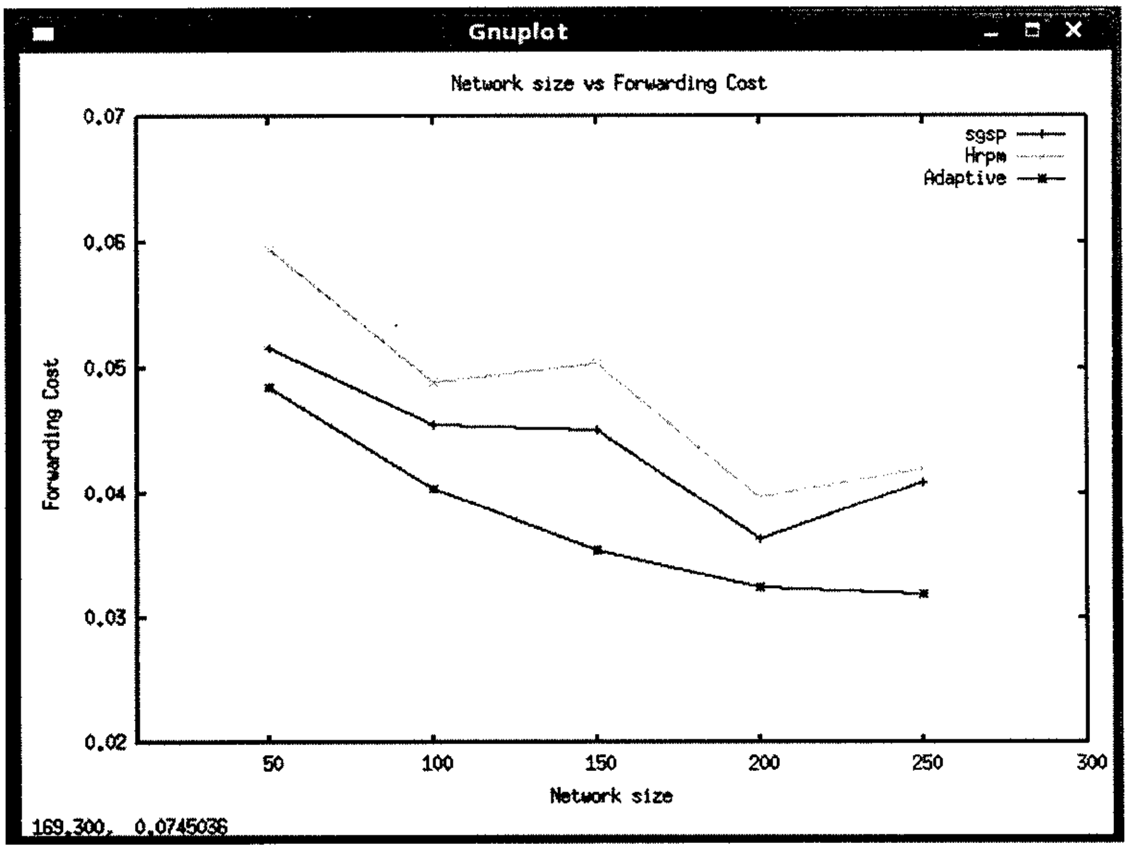
Inference:

From the above graph it is inferred that adaptive algorithm provides about 84.41% of minimized overhead which is comparatively efficient than SGSP giving 74.62% minimized overhead and HRPM giving 79.41% minimized overhead. Also Adaptive algorithm is 17.43% higher than HRPM and 10.84% higher than SGSP.

Thus the adaptive algorithm is efficient because in adaptive coordinator election algorithm the leader nodes are elected based on nodes average speed.

- **Speed vs Forwarding cost:**

The graph is plotted by considering speed (50, 100, 150, 200, and 250) on x-axis and Forwarding cost on y-axis. The Forwarding cost of nodes is analyzed by changing Pause time (5, 10, 15, 20, and 25).



Inference:

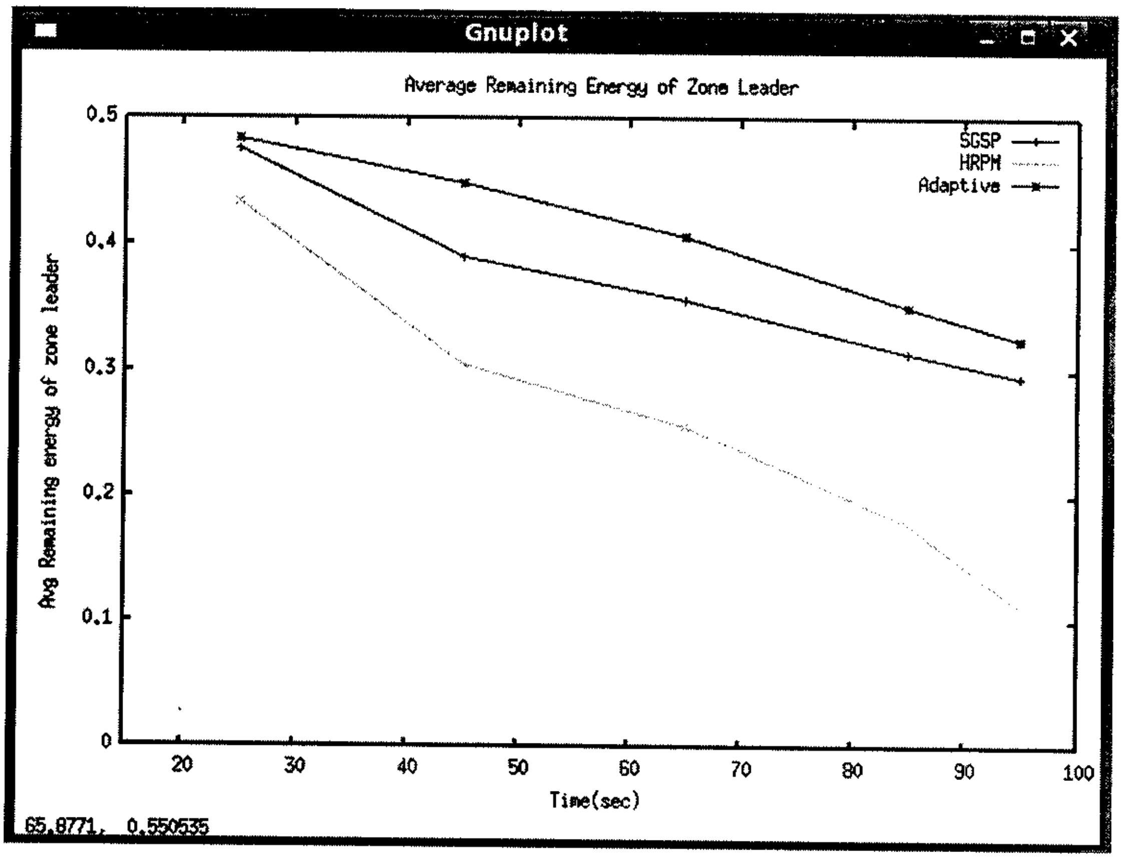
From the above graph it is inferred that adaptive algorithm provides about 82.92% of less forwarding cost which is comparatively efficient than SGSP giving 73.44% less forwarding cost and HRPM giving 78.47% less forwarding cost. Also Adaptive algorithm is 21.62% higher than HRPM and 14.12% higher than SGSP.

Thus the adaptive algorithm is efficient because in adaptive coordinator election algorithm the leader nodes are elected based on nodes average speed.

c) Impact on Remaining Energy of the Zone Leader:

The graph is plotted by considering network size (15, 20, 30, 40, 50, and 60) on x-axis and Average Zone Leader Change on y-axis.

The Remaining Energy of the Zone Leader is analyzed by considering remaining energy of zone leader for at time (25, 45, 65, 85, and 95).



Inference:

From the above graph it is inferred that adaptive algorithm provides about 86.77% of remaining energy which is comparatively efficient than SGSP giving 78.41% less forwarding cost and HRPM giving 71.62% less forwarding cost. Also Adaptive algorithm is 16.93% higher than HRPM and 9.89% higher than SGSP.

Because the leaders changes only if the eligibility factor existing Zone Leader is less than the threshold limit, and this eligibility factor mainly depends on the zone leaders remaining energy.

Thus the zone leader in adaptive leader election algorithm holds as leader for longer time which is advantageous in minimizing the overhead.

CHAPTER 6
CONCLUSION AND FUTURE WORK

A scalable and efficient service coordination scheme for MANET has been proposed. The scheme is adaptable to the changing topology and management requirements of the network. The system also accomplishes a scalable resource tracking, service membership management mechanism to support timely and coordinative service provisioning.

It has compared and analyzed the three algorithms for the service coordinator election and has arrived at the conclusion justifying the efficiency of the adaptive leader election algorithm and the proficient routing of data packets through the leader elected by that algorithm. The justification is supported evidently by the performance evaluation of impact on network size with Average Zone Leader Change, also with Average No. of Zone Leader, with impact on Effect of Moving Speed of Zone Leader and impact on Remaining Energy of Zone Leader.

The system is more scalable in terms of number of multicast groups compared to other protocols. Still the future research can be made in the following directions:

(a) More efficient schemes for service discovery to be considered supporting better resource tracking.

(b) Intend to maintain a proper group membership management and Location updates.

APPENDICES

Source Code

```
# initial configuration
set val(chan)           Channel/WirelessChannel              ;#Channel Type
set val(prop)           Propagation/TwoRayGround         ;# radio-propagation model
set val(netif)          Phy/WirelessPhy                   ;# network interface type
set val(mac)            Mac/802_11                                   ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue              ;# interface queue type
set val(ll)             LL                                     ;# link layer type
set val(ant)            Antenna/OmniAntenna                     ;# antenna model
set val(ifqlen)         15                                  ;# max packet in ifq
set val(nn)             15                             ;# number of mobilenodes
set val(rp)             AODV                                ;# routing protocol
set val(x)              2400                      ;# x dimension of topography
set val(y)              2400                      ;# y dimension of topography
set val(seed)           2.0
set val(stop)           100                          ;#time of simulation end
set val(sc)             "output-15-scen"
set val(intvallocal)    4
set val(intvalglobal)   6
set val(energymodel)    EnergyModel
set val(initialenergy)  0.5

# Initialize network simulator
set ns_         [new Simulator]

# tracefile creation
set tracefile1    [open 15-m-idbased.tr w]
$ns_ trace-all $tracefile1

# channel #1 and #2 creation
set chan_1_ [new $val(chan)]

# nam creation
set namfile [open 15-m-idbased.nam w]
$ns_ namtrace-all-wireless $namfile $val(x) $val(y)
# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Create god
set god_ [create-god $val(nn)]

#setting color for data flow
```

```
$ns_ color 0 blue
# node configuration
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace OFF \
                 -channel $chan_1_\
                -energyModel $val(energymodel)\
                -rxPower 0.3\
                -txPower 0.6\
                -initialEnergy $val(initialenergy)\
                -batteryModel RTBattery\
                -alpha 35220\
                -beta 0.637\
                -voltage 4.1
# creation of nodes
for { set i 0 } { $i < 15 } { incr i } {
 global node_
 set node_($i) [$ns_ node]

 #disable random motion
 $node_($i) random-motion 0
}
# loading the initial positions and node movements
puts "Loading scenario pattern..."
source $val(sc)

# create inititial node position in nam
#size of node is 15
for { set i 0} { $i < 15 } {incr i} {
  $ns_ initial_node_pos $node_($i) 15
}
#origin
set x0 0.00
set y0 0.00

#total number of nodes
```

```
global tot_nodes
set tot_nodes $val(nn)

#buffer node for service cache
set node_(15) [$ns_ node]
$ns_ initial_node_pos $node_(15) 15
$node_(15) set speed_ 20
#setting frequency for the nodes
for { set i 0 } { $i < 15 } { incr i } {
 $node_($i) set freq 0
 $node_($i) set type 3
}
#assigning every 4th node as a service provider
global sp
set j 0
for {set i 0} {$i < 4} {incr i} {
 set sp($i) $j
 set j [expr $j+4]
 }
#service provisioning
for {set i 0} {$i < 15} {incr i} {
 $node_($i) set type 2
 $node_($i) set sr_count 0
 set i [ expr $i + 3 ]
}

$node_(0) set services(0) 1
$node_(0) set services(1) 5
$node_(0) set services(2) 9
$node_(0) set services(3) 13
$node_(0) set services(4) 17
$node_(0) set service_count 5

$node_(4) set services(0) 2
$node_(4) set services(1) 6
$node_(4) set services(2) 10
$node_(4) set services(3) 14
$node_(4) set services(4) 18
$node_(4) set service_count 5

#setting lifetime for services
$node_(0) set life_time(0) 80
$node_(0) set life_time(1) 140
$node_(0) set life_time(2) 270
$node_(0) set life_time(3) 400
$node_(0) set life_time(4) 130
```

```
$node_(4) set life_time(0) 90
$node_(4) set life_time(1) 115
$node_(4) set life_time(2) 280
$node_(4) set life_time(3) 70
$node_(4) set life_time(4) 120

#data structure that stores the access points in each zone
global leader
set lptr 0
#global param
global count1 count2 count3 count4 count5 count6 count7 count8 count9
global back_up1 back_up2 back_up3 back_up4 back_up5 back_up6 back_up7 back_up8
back_up9
global cx cy
global a b
global count5
global rp
global initx inity id za zb inita initb
set rp 0
#storing initial positions
for { set i 0 } { $i < 15 } { incr i } {
 set initx($i) "[ $node_($i) set X_ ]"
 set inity($i) "[ $node_($i) set Y_ ]"
 set id($i)  $node_($i)
 set inita($i) [ expr abs(floor(("[ $node_($i) set X_ ]" - $x0 ) / 800.0))]
 set initb($i) [ expr abs(floor(("[ $node_($i) set Y_ ]" - $y0 ) / 800.0))]
}

#virtual zone construction
proc zone_construct { tz } {
 global ns_ node_ x y x0 y0 cx cy tot_nodes
 global a b
puts " -------------------------------------------------------------------------"
puts "                    VIRTUAL ZONE CONSTRUCTION"
puts " -------------------------------------------------------------------------"
# calculation of a and b which form the zone id
 for {set i 0} {$i < $tot_nodes} {incr i} {
 set a($i) [ expr abs(floor(("[ $node_($i) set X_ ]" - $x0 ) / 800.0))]
  $node_($i) set a $a($i)
  $node_($i) set zid(0) $a($i)
  set b($i) [ expr abs(floor(("[ $node_($i) set Y_ ]" - $y0 ) / 800.0))]
  $node_($i) set b $b($i)
  $node_($i) set zid(1) $b($i)
  #$ns_ at $tz  "puts [ $node_($i) set en ]"
 }
```

```
#calculating centre of each zone
 set temp 0
 for {set i 0} {$i < 3} {incr i} {
   for {set j 0} {$j < 3} {incr j} {
    set cx($temp)  [ expr ( $x0 + (( $i + 0.5 ) * 800.0 )) ]
    set cy($temp)  [ expr ( $y0 + (( $j + 0.5 ) * 800.0 )) ]
    set temp [ expr $temp + 1 ]
    }
   }
#parameters containing the number of nodes in each zone and the zone array
 global r1 r2 r3 r4 r5 r6 r7 r8 r9 z1 z2 z3 z4 z5 z6 z7 z8 z9 index lead s
 set r1 0
 for {set i 0} {$i < $tot_nodes} {incr i} {

  if { $a($i) == 0 && $b($i) == 0 } {
         set z1($r1) $i
         $node_($i) set zn 1
         set r1 [ expr $r1 + 1 ]
  }
  if { $a($i) == 0 && $b($i) == 1 } {
         set z2($r2) $i
         $node_($i) set zn 2
         set r2 [ expr $r2 + 1 ]
  }
  if { $a($i) == 2 && $b($i) == 2 } {
         set z9($r9) $i
         $node_($i) set zn 9
         set r9 [ expr $r9 + 1 ]
  }
 }

#displaying the nodes in each zone
puts "Nodes in Zone 1"
puts "Number of nodes in zone1 : $r1"
 for {set i 0} {$i < $r1} {incr i} {
puts "$z1($i)"
 }
}

#zone1 - leader election and service registration
proc elect1 {ts1 te1 } {
 global ns_
 global node_
 global z1 r1
 global sp
 global leader lptr cx cy
 set j 0
 global back_up1 velocity frequency ef

 #ID BASED LEADER ELECTION FOR ZONE 1
 #node id
 if { $r1 > 1 } {
  for {set i 0} {$i < $r1} {incr i} {
   set index $z1($i)
   set nid($i) $node_($index)
  }
  #sorting id
  for {set i 0} {$i < $r1} {incr i} {
   set node1($i) $z1($i)
  }
  for {set i 0} {$i < $r1} {incr i} {
   for {set j 1} {$j < $r1} {incr j} {
    if { $nid($i) < $nid($j) } {
     set temp1 $nid($i)
     set nid($i) $nid($j)
     set nid($j) $temp1
     set temp2 $node1($i)
     set node1($i) $node1($j)
     set node1($j) $temp2
    }
   }
 #puts "$nid($i)"
 }
 #leader election
  set lead $node1(0)
  set back_up1 $node1(1)
}

 if { $r1 == 1 } {
 set lead $z1(0)
 }
 if { $r1 == 0 } {
 set lead 15
 }
set leader($lptr) $lead

$node_($lead) set type 1
$node_($lead) set mem_count 0
 $node_($lead) set lc $lead
set lptr [expr $lptr + 1]
if { $lead != 15 } {
$ns_ at $ts1 "$ns_ trace-annotate \" node $lead is elected as the AP for zone 1 \""
```

```
 for { set i 0 } { $i  < $r1 } { incr i } {
   if { $z1($i) != $leader(0) } {
    set index $leader(0)
    set udp_($index) [new Agent/UDP]
    $ns_ attach-agent $node_($index) $udp_($index)
    set null_($index) [new Agent/Null]
    $ns_ attach-agent $node_($z1($i)) $null_($index)
    set cbr_($index) [new Application/Traffic/CBR]
    $cbr_($index) set packetSize_ 512
    $cbr_($index) set interval_ 4.0
    $cbr_($index) set random_ 1
    $cbr_($index) set maxpkts_ 10000
    $cbr_($index) attach-agent $udp_($index)
    $ns_ connect $udp_($index) $null_($index)
    $ns_ at $ts1 "$cbr_($index) start"
    $ns_ at $te1 "$cbr_($index) stop"
    $ns_ at $ts1 "$ns_ trace-annotate \"AP NODE $leader(0) announces itself as leader to
node $z1($i) of zone 1 \""
    $node_($z1($i)) set lc $index
    }
   }
  }
puts "Leader of Zone1: $lead "
}

# Election of rendezvous point
proc elect_rp { time } {
 global ns_ node_
 global s
 global leader
 global rp
 set maxval $node_($leader(0))
 for {set i 1} {$i < 9} {incr i} {
  set j $leader($i)
  if {$node_($j) > $maxval} {
   set maxval $node_($j)
   set rp $j
  }
 }
 $node_($rp) set type 0
 $node_($rp) set freq [ expr "[ $node_($rp) set freq ]" + 1 ]
# puts "Node $rp is the current rp"
}

proc update_table_ap { time1 time2 } {
 global count1 count2 count3 count4 count5 count6 count7 count8 count9
global z1 z2 z3 z4 z5 z6 z7 z8 z9
global r1 r2 r3 r4 r5 r6 r7 r8 r9
global ns_ node_
global sp a b
global count lptr
global leader
#SERVICE REGISTRATION FOR ZONE 1
if { $leader(0) != 15 } {
for {set i 0} {$i < $r1} {incr i} {
 if { $z1($i) != $leader(0) } {
  for {set k 0} {$k <4} {incr k} {
   if { $z1($i) == $sp($k) } {
   set index $z1($i)
   set udp_($index) [new Agent/UDP]
   $ns_ attach-agent $node_($index) $udp_($index)
   set null_($index) [new Agent/Null]
   $ns_ attach-agent $node_($leader(0)) $null_($index)
   set cbr_($index) [new Application/Traffic/CBR]
   $cbr_($index) set packetSize_ 512
   $cbr_($index) set interval_ 4.0
   $cbr_($index) set random_ 1
   $cbr_($index) set maxpkts_ 10000
   $cbr_($index) attach-agent $udp_($index)
   $ns_ connect $udp_($index) $null_($index)
   $ns_ at $time1 "$cbr_($index) start"
   $ns_ at $time2 "$cbr_($index) stop"
   $ns_ at $time1 "$ns_ trace-annotate \"SP NODE $index REGISTERS its SERVICES
WITH AP NODE $leader(0) of zone 1 \""

   set time1 [expr $time1 + 0.02]
   set time2 [expr $time2 + 0.02]
   }}}}}
#storing information of access point 1
set count1 0
if { $leader(0) == 15 } {
  $node_($leader(0)) set sp($count1) 15
  $node_($leader(0)) set sp_loc($count1,0) $ns_ at $time1 "[ $node_(15) set X_ ]"
  $node_($leader(0)) set sp_loc($count1,1) $ns_ at $time1 "[ $node_(15) set Y_ ]"
  $node_($leader(0)) set sp_services($count1,0) 0
  $node_($leader(0)) set sp_count $count1
} else {
 for {set m 0} {$m < $r1 } {incr m} {
   if { $z1($m) % 4 == 0 } {
  #storing the service providers of ap1
   $node_($leader(0)) set sp($count1) $z1($m)
  #storing the locations
```

```
   $node_($leader(0)) set sp_loc($count1,0) $ns_ at $time1 "[ $node_($z1($m)) set X_ ]"
   $node_($leader(0)) set sp_loc($count1,1) $ns_ at $time1 "[ $node_($z1($m)) set Y_ ]"

  #storing all the services
  $node_($leader(0)) set sp_services_count($count1) " [ $node_($z1($m)) set
service_count ] "
   for { set i 0 } { $i < " [ $node_($z1($m)) set service_count ] " } {incr i } {

     $node_($leader(0)) set sp_services($count1,$i) "[$node_($z1($m)) set services($i) ]"
     $node_($leader(0)) set sp_services_life($count1,$i) "[$node_($z1($m)) set
life_time($i) ]"
   }
   set count1 [expr $count1 + 1]
  }}}
 $node_($leader(0)) set sp_count $count1

proc update_table_rp { tr } {
 global node_ ns_
 global rp leader
 set temp 0
 for { set k 0 } { $k < 9 } { incr k } {
  set spcount 0
  if { $leader($k) != 15 } {
     if {$leader($k) != $rp} {
     set index $rp
    set index2 $leader($k)
    set udp_($index) [new Agent/UDP]
    $ns_ attach-agent $node_($index) $udp_($index)
    set null_($index) [new Agent/Null]
    $ns_ attach-agent $node_($index2) $null_($index)
    set cbr_($index) [new Application/Traffic/CBR]
    $cbr_($index) set packetSize_ 512
    $cbr_($index) set interval_ 4.0
    $cbr_($index) set random_ 1
    $cbr_($index) set maxpkts_ 10000
    $cbr_($index) attach-agent $udp_($index)
    $ns_ connect $udp_($index) $null_($index)
    $ns_ at $tr "$cbr_($index) start"
    $ns_ at [expr $tr + 0.02] "$cbr_($index) stop"
    $ns_ at $tr "$ns_ trace-annotate \"RP NODE $index announces itself as RP to the AP
$index2 \""
  }
  set temp1 " [ $node_($leader($k)) set sp_count ] "
  $node_($rp) set ap($k) $leader($k)
  $node_($rp) set ap_zid($k,0) "[ $node_($leader($k)) set zid(0) ]"
  $node_($rp) set ap_zid($k,1) "[ $node_($leader($k)) set zid(1) ]"
  for { set i 0 } { $i < $temp1 } { incr i } {
    set temp2 " [ $node_($leader($k)) set sp_services_count($i) ] "
    for { set j 0 } { $j < $temp2 } { incr j } {
      $node_($rp) set ap_services($k,$spcount) " [ $node_($leader($k)) set
sp_services($i,$j) ] "
      set spcount [ expr $spcount + 1 ]
     }}}
 $node_($rp) set ap_service_count($k) $spcount
 }}

proc refresh { tf } {
 global ns_ sp node_ leader
 for {set i 0} {$i < 3} {incr i} {
  set index $sp($i)
  for {set j 0} {$j < " [ $node_($index) set service_count ] "} {incr j} {
   if { " [ $node_($index) set life_time($j) ] " != 0 } {
    if { [ expr " [ $node_($index) set life_time($j) ] " % $tf ] == 0} {
     set temp [ expr " [ $node_($index) set zn ] " - 1 ]
     set udp_($index) [new Agent/UDP]
     $ns_ attach-agent $node_($index) $udp_($index)
     set null_($index) [new Agent/Null]
     $ns_ attach-agent $node_($leader($temp)) $null_($index)
     set cbr_($index) [new Application/Traffic/CBR]
     $cbr_($index) set packetSize_ 512
     $cbr_($index) set interval_ 4.0
     $cbr_($index) set random_ 1
     $cbr_($index) set maxpkts_ 10000
     $cbr_($index) attach-agent $udp_($index)
     $ns_ connect $udp_($index) $null_($index)
     $ns_ at $tf "$cbr_($index) start"
     set end [ expr $tf + 0.02 ]
     $ns_ at $end "$cbr_($index) stop"
     $ns_ at $tf "$ns_ trace-annotate \"SP NODE $index REFRESHES its SERVICE $j
WITH AP NODE $leader($temp) of zone [ expr $temp + 1 ] \""
    }}}}}

proc add_sp {tm } {
 global ns_ node_
 global leader count5 ser5 ap5_ser z5 r5
 set te [expr $tm + 0.05]
 set temp1 [expr $r5 - 1]
 set temp2 $z5($temp1)
 if { $temp2 % 4 == 0 } {
  set temp1 [expr $temp1 - 1]
  set temp2 $z5($temp1)
  }
 set scount "[ $node_($leader(4)) set sp_count ]"
 $node_($temp2) set type 2
 $node_($temp2) set services(0) 4
 $node_($temp2) set services(1) 5
 $node_($temp2) set services(2) 12
 $node_($temp2) set life_time(2) 115
 $node_($temp2) set life_time(3) 170
 $node_($temp2) set service_count 4
  #registering services with ap of zone $temp
   set udp_($temp2) [new Agent/UDP]
   $ns_ attach-agent $node_($temp2) $udp_($temp2)
   set null_($temp2) [new Agent/Null]
   $ns_ attach-agent $node_($leader(4)) $null_($temp2)
   set cbr_($temp2) [new Application/Traffic/CBR]
   $cbr_($temp2) set packetSize_ 512
   $cbr_($temp2) set interval_ 4.0
   $cbr_($temp2) set random_ 1
   $cbr_($temp2) set maxpkts_ 10000
   $cbr_($temp2) attach-agent $udp_($temp2)
   $ns_ connect $udp_($temp2) $null_($temp2)
   $ns_ at $tm "$cbr_($temp2) start"
   $ns_ at $te "$cbr_($temp2) stop"
   $ns_ at $tm "$ns_ trace-annotate \"A new SP NODE $temp2 REGISTERS its
SERVICES WITH AP NODE $leader(4) of zone 5 \""
 #updating ap table
   $node_($leader(4)) set sp($scount) $temp2
   $node_($leader(4)) set sp_count [expr $scount + 1]
   $node_($leader(4)) set sp_loc($scount,0) $ns_ at $tm "[ $node_($temp2) set X_ ]"
   $node_($leader(4)) set sp_loc($scount,1) $ns_ at $tm "[ $node_($temp2) set Y_ ]"
   $node_($leader(4)) set sp_services($scount,0) 4
   $node_($leader(4)) set sp_services($scount,1) 5
   $node_($leader(4)) set sp_services_life($scount,2) 115
   $node_($leader(4)) set sp_services_life($scount,3) 170
    $node_($leader(4)) set sp_services_count($scount) 4
   set $count5 [ expr $count5 + 1 ]
}

proc del_sp { td } {
 global ns_ node_ leader count2 ser2 ap2_ser
 set ind [ expr $count2 - 1 ]
 set temp2 "[$node_($leader(1)) set sp($ind)]"
 $node_($temp2) set service_count 0
 $node_($temp2) set type 3
 for { set i 0 } { $i < 5 } { incr i } {
  $node_($temp2) set services($i) 0
  $node_($temp2) set life_time($i) 0
 }
 set temp [ expr " [ $node_($leader(1)) set sp_count ] " - 1 ]
 $node_($leader(1)) set sp_count $temp
  set count2 [ expr $count2 - 1 ]
 $ns_ at $td "$ns_ trace-annotate \"SP NODE $temp2 DELETES its SERVICES FROM
AP NODE $leader(1) of zone 2 \""
 }

proc ap_fail { timef } {
 global ns_ node_ back_up4 leader z4 r4 rp
 $ns_ at $timef "$ns_ trace-annotate \"AP NODE $leader(3) of zone 4 fails due to system
crash \""
 if { $leader(3) % 4 == 0 } {
  $node_($leader(3)) set type 2
 } else {
  $node_($leader(3)) set type 3
 }
 $node_($leader(3)) set speed_ 0
 set $leader(3) $back_up4
 for { set i 0 } { $i  < $r4 } { incr i } {
   if { $z4($i) != $leader(3) } {
    set index $leader(3)
    set udp_($index) [new Agent/UDP]
    $ns_ attach-agent $node_($index) $udp_($index)
    set null_($index) [new Agent/Null]
    $ns_ attach-agent $node_($z4($i)) $null_($index)
    set cbr_($index) [new Application/Traffic/CBR]
    $cbr_($index) set packetSize_ 512
    $cbr_($index) set interval_ 4.0
    $cbr_($index) set random_ 1
    $cbr_($index) set maxpkts_ 10000
    $cbr_($index) attach-agent $udp_($index)
    $ns_ connect $udp_($index) $null_($index)
    $ns_ at $timef "$cbr_($index) start"
    $ns_ at [ expr $timef + 0.05 ] "$cbr_($index) stop"
    $ns_ at $timef "$ns_ trace-annotate \"BACK UP NODE $leader(3) announces itself as
AP to node $z4($i) of zone 4 \""
    $node_($z4($i)) set lc $index
   }
 }
 $node_($leader(3)) set type 1
 $node_($leader(3)) set mem_count 0
 $node_($leader(3)) set freq [ expr "[ $node_($leader(3)) set freq ]" + 1 ]
    set index $leader(3)
    set udp_($index) [new Agent/UDP]
    $ns_ attach-agent $node_($index) $udp_($index)
   set null_($index) [new Agent/Null]
   $ns_ attach-agent $node_($rp) $null_($index)
   set cbr_($index) [new Application/Traffic/CBR]
   $cbr_($index) set packetSize_ 512
   $cbr_($index) set interval_ 4.0
   $cbr_($index) set random_ 1
   $cbr_($index) set maxpkts_ 10000
   $cbr_($index) attach-agent $udp_($index)
   $ns_ connect $udp_($index) $null_($index)
   $ns_ at $timef "$cbr_($index) start"
   $ns_ at [ expr $timef + 0.05 ] "$cbr_($index) stop"
   $ns_ at $timef "$ns_ trace-annotate \"BACK UP NODE $leader(3) updates itself with
the RP NODE $rp \""
   $node_($rp) set ap(3) $leader(3)
}

#neighbor table creation
proc update_neighbor { timen } {
 global a b node_ ns_
 for { set i 0 } { $i < 15 } {incr i} {
   set cnt 0
   set tx1 "[ $node_($i) set X_ ]"
   set ty1 "[ $node_($i) set Y_ ]"
  for { set j 0 } {$j < 15 } {incr j} {
    set tx2 "[ $node_($j) set X_ ]"
    set ty2 "[ $node_($j) set Y_ ]"
    set xd  [ expr $tx1 - $tx2 ]
    set yd [ expr $ty1 - $ty2 ]
    set d [ expr sqrt ( ( $xd * $xd ) + ( $yd * $yd ) ) ]
    if { $d < 251 && $d != 0} {
     $ns_ at $timen "$ns_ trace-annotate \" Neighbor $j is updated in $i 's neighbor table\"
"
     $node_($i) set nid($cnt) $j
     $node_($i) set n_ap($cnt) "[ $node_($j) set lc ]"
     $node_($i) set pos($cnt,0) $tx2
     $node_($i) set pos($cnt,1) $ty2
     $node_($i) set n_zid($cnt,0) "[ $node_($j) set a ]"
     $node_($i) set n_zid($cnt,1) "[ $node_($j) set b ]"
     if { "[ $node_($j) set type ]" == 1 } {
      $node_($i) set flag($cnt) 1
     } else {
         $node_($i) set flag($cnt) 0
     }
    set cnt [ expr $cnt + 1 ]
   } }
  $node_($i) set ncount $cnt
 }}

#neighbor table updation
proc hello_beacon { timeb1 timeb2} {
 global node_ ns_ initx inity id inita initb
 for { set i 0 } { $i < 15 } { incr i } {
 set newx($i) "[ $node_($i) set X_ ]"
 set newy($i) "[ $node_($i) set Y_ ]"
 }
 for { set i 0 } { $i < 15 } { incr i } {
   set xd1  [ expr $initx($i) - $newx($i) ]
   set yd1 [ expr $inity($i) - $newy($i) ]
   set d1 [ expr sqrt ( ( $xd1 * $xd1 ) + ( $yd1 * $yd1 ) ) ]
   if { $d1 > 215 } {
    set initx($i) $newx($i)
    set inity($i) $newy($i)
    set ncnt "[ $node_($i) set ncount ]"
    for { set j 0 } { $j < $ncnt } { incr j } {
     set index $i
     set index2 "[ $node_($i) set nid($j) ]"
     set udp_($index) [new Agent/UDP]
     $ns_ attach-agent $node_($index) $udp_($index)
     set null_($index) [new Agent/Null]
     $ns_ attach-agent $node_($index2) $null_($index)
     set cbr_($index) [new Application/Traffic/CBR]
     $cbr_($index) set packetSize_ 512
     $cbr_($index) set interval_ 4.0
     $cbr_($index) set random_ 1
     $cbr_($index) set maxpkts_ 10000
     $cbr_($index) attach-agent $udp_($index)
     $ns_ connect $udp_($index) $null_($index)
     $ns_ at $timeb1 "$cbr_($index) start"
     $ns_ at [expr $timeb1 + 0.02] "$cbr_($index) stop"
     $ns_ at $timeb1 "$ns_ trace-annotate \"NODE $index sends hello message to $index2
\""
         }}}}
proc rp_move { timer } {
 global node_ ns_ initx inity id inita initb rp
 set newxrp "[ $node_($rp) set X_ ]"
 set newyrp "[ $node_($rp) set Y_ ]"
 set xd1  [ expr $initx($rp) - $newxrp ]
 set yd1  [ expr $inity($rp) - $newyrp ]
 set d1 [ expr sqrt ( ( $xd1 * $xd1 ) + ( $yd1 * $yd1 ) ) ]
   if { $d1 > 100 } {
    set initx($rp) $newxrp
    set inity($rp) $newyrp
```

```
     for { set j 0 } { $j < 9 } { incr j } {
      set index $rp
      set index2 "[ $node_($rp) set ap($j) ]"
      set udp_($index) [new Agent/UDP]
      $ns_ attach-agent $node_($index) $udp_($index)
      set null_($index) [new Agent/Null]
      $ns_ attach-agent $node_($index2) $null_($index)
      set cbr_($index) [new Application/Traffic/CBR]
      $cbr_($index) set packetSize_ 512
      $cbr_($index) set interval_ 4.0
      $cbr_($index) set random_ 1
      $cbr_($index) set maxpkts_ 10000
      $cbr_($index) attach-agent $udp_($index)
      }}}
proc new_node { timex timey } {
 global ns_ node_ leader inita initb
 for { set i 0 } { $i < 15 } { incr i } {
   if {"[ $node_($i) set a]" != $inita($i) || "[ $node_($i) set b]" != $initb($i) } {
    set ncnt "[ $node_($i) set ncount ]"
    for { set j 0 } { $j < $ncnt } { incr j } {
    if { ("[ $node_($i) set a ]" == "[ $node_($i) set n_zid($j,0) ]") && ("[ $node_($i) set b
]" == "[ $node_($i) set n_zid($j,1) ]") } {
     set index $i
     set index2 "[ $node_($i) set nid($j) ]"

     set udp_($index) [new Agent/UDP]
     $ns_ attach-agent $node_($index) $udp_($index)
     set null_($index) [new Agent/Null]
     $ns_ attach-agent $node_($index2) $null_($index)
     set cbr_($index) [new Application/Traffic/CBR]
     $cbr_($index) set packetSize_ 512
     $cbr_($index) set interval_ 4.0
     $cbr_($index) set random_ 1
     $cbr_($index) set maxpkts_ 10000
     $cbr_($index) attach-agent $udp_($index)
     $ns_ connect $udp_($index) $null_($index)
     $ns_ at $timex "$cbr_($index) start"
     $ns_ at [expr $timex + 0.02] "$cbr_($index) stop"
     $ns_ at $timex "$ns_ trace-annotate \"NODE $index sends hello message to its
neighbor $index2 \""
          }}}}}

proc service_discovery { timed sr serv} {
  global ns_ node_ leader rp
    #SR sends its request to its AP
    set access_point "[ $node_($sr) set lc]"
```

```
      set index $access_point
      set index2 $sr
      set udp_($index2) [new Agent/UDP]
      $ns_ attach-agent $node_($index2) $udp_($index2)
      set null_($index2) [new Agent/Null]
      $ns_ attach-agent $node_($index) $null_($index2)
      set cbr_($index2) [new Application/Traffic/CBR]
      $cbr_($index2) set packetSize_ 512
      $cbr_($index2) set interval_ 4.0
      $cbr_($index2) set random_ 1
      $cbr_($index2) set maxpkts_ 10000
      $cbr_($index2) attach-agent $udp_($index2)
      $ns_ connect $udp_($index2) $null_($index2)
      $ns_ at $timed "$cbr_($index2) start"
      $ns_ at [expr $timed + 0.01] "$cbr_($index2) stop"
  $ns_ at $timed "$ns_ trace-annotate \"SR NODE $sr sends a request message to its AP
NODE $index \""
  set timed [expr $timed + 0.01]
  set cn "[ $node_($access_point) set mem_count]"
  $node_($access_point) set sr_id($cn) $sr
  set flag 0
  #if service available in the requestor table
  if {  "[ $node_($access_point) set mem_count]" > 0 } {
   for { set i 0 } { $i < "[ $node_($access_point) set mem_count]" } {incr i } {
    if { $serv ==  "[ $node_($access_point) set service_id($i) ] "  } {
     set flag 1
     #AP forwards the request to the SP
     $ns_ at $timed "$ns_ trace-annotate \" Service $serv available in the AP NODE
$access_point (requestor table)  \""
      set index "[ $node_($access_point) set sp_id($i) ]"
      set index2 $access_point
      set udp_($index2) [new Agent/UDP]
      $ns_ attach-agent $node_($index2) $udp_($index2)
      set null_($index2) [new Agent/Null]
      $ns_ attach-agent $node_($index) $null_($index2)
      set cbr_($index2) [new Application/Traffic/CBR]
      $cbr_($index2) set packetSize_ 512
      $cbr_($index2) set interval_ 4.0
      $cbr_($index2) set random_ 1
      $cbr_($index2) set maxpkts_ 10000
      $cbr_($index2) attach-agent $udp_($index2)
      $ns_ connect $udp_($index2) $null_($index2)
      $ns_ at $timed "$cbr_($index2) start"
      $ns_ at [expr $timed + 0.01] "$cbr_($index2) stop"
     $ns_ at $timed "$ns_ trace-annotate \"AP NODE $index2 forwards the request
message to SP NODE $index \""
```

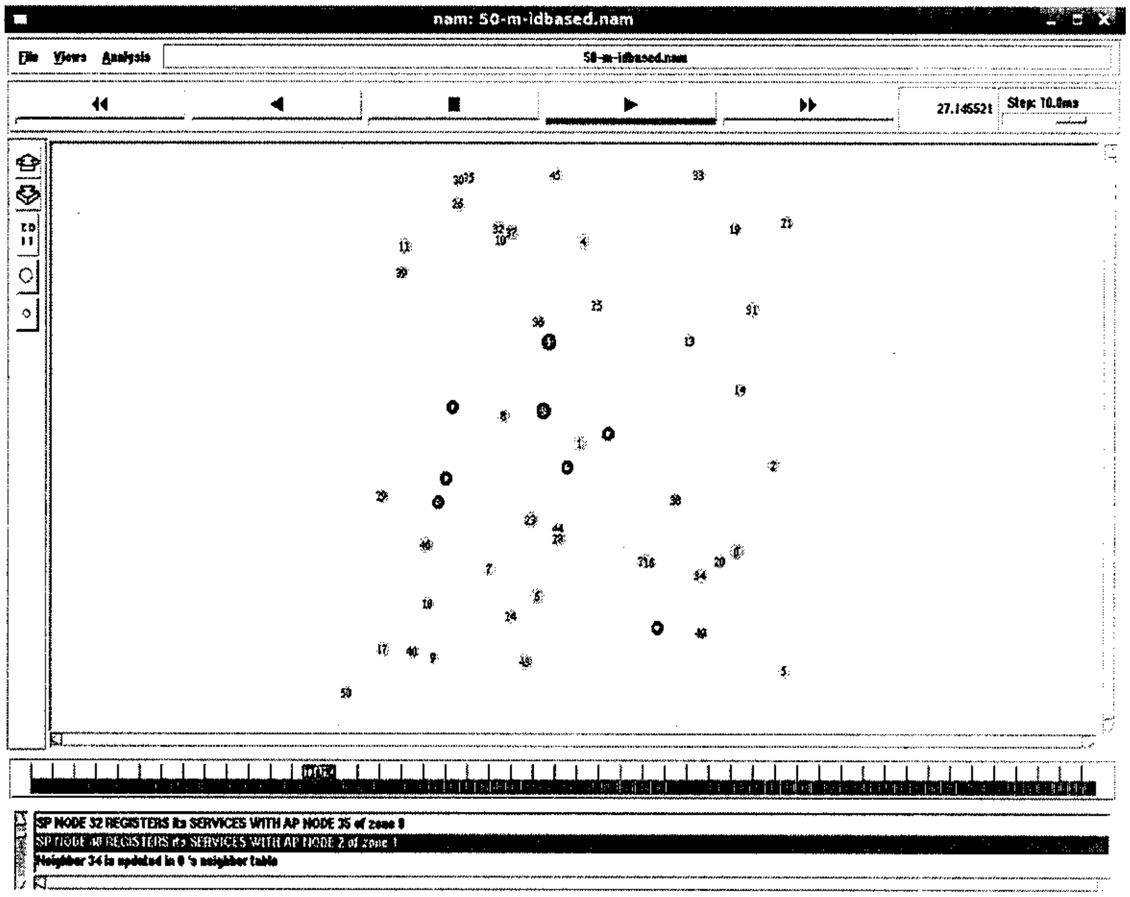
```
    set timed [expr $timed + 0.01]
    $node_($access_point) set sr_id($i) $sr
   }}}
  # service available within the same zone
  if { $flag != 1} {
  set n " [ $node_($access_point) set sp_count ] "
  set cnt 0
  for { set i 0 } { $i < $n } { incr i } {
   set k "[ $node_($access_point) set sp_services_count($i)]"
   for { set j 0 } { $j < $k } { incr j } {
     if { $serv ==  "[ $node_($access_point) set sp_services($i,$j) ] "  } {
         set flag 2
            set temp_arr($cnt) $i
         set temp_id($cnt) "[ $node_($access_point) set sp($i)]"
         set temp_life($cnt) "[ $node_($access_point) set sp_services_life($i,$j) ] "
            set cnt [expr $cnt + 1]
        }}}
  if { $flag == 2 } {
     for { set i 0 } { $i < $cnt } {incr i} {
 set xdiff [ expr "[$node_($sr) set X_]" - "[ $node_($temp_id($i)) set X_ ]" ]
  set ydiff [ expr "[$node_($sr) set Y_]" - "[ $node_($temp_id($i)) set Y_ ]" ]
   set dist($i) [ expr sqrt ( ( $xdiff * $xdiff ) + ( $ydiff * $ydiff ) ) ]
     }
     set period 0.04
     set min_dist 1000
     for { set i 0 } { $i < $cnt } {incr i} {
          if { $temp_life($i) > $period && $dist($i) < $min_dist } {
           set min $temp_arr($i)
        set min_dist $dist($i)
            } }
     if { $flag == 3 } {
          for { set i 0 } { $i < $cnt } {incr i} {
  set xdiff [ expr "[$node_($sr) set X_]" - "[ $node_($temp_id($i)) set X_ ]" ]
  set ydiff [ expr "[$node_($sr) set Y_]" - "[ $node_($temp_id($i)) set Y_ ]" ]
  set dist($i) [ expr sqrt ( ( $xdiff * $xdiff ) + ( $ydiff * $ydiff ) ) ]
       }
       set min_dist 1000
       for { set i 0 } { $i < $cnt } {incr i} {
         if { $dist($i) < $min_dist } {
               set min $temp_arr($i)
            set min_dist $dist($i)
            }}
       if { $cnt == 1 } {
         set temp $temp_arr(0)
       } else  {
         set temp $min
        }
        set period 0.04
        set min $temp1_arr(0)
        for { set i 0 } { $i < $cnt } {incr i} {
          if { $temp1_life($i) > $period} {
                set min $temp1_arr($i)
        }}
        if { $cnt == 1 } {
          set temp1 $temp1_arr(0)
        } else {
          set temp1 $min
        }
        #AP forwards the request to the SP
        set index "[ $node_($leader($temp)) set sp($temp1) ]"
        set index2 $leader($temp)
            set udp_($index2) [new Agent/UDP]
            $ns_ attach-agent $node_($index2) $udp_($index2)
            set null_($index2) [new Agent/Null]
        $ns_ attach-agent $node_($index) $null_($index2)
        set cbr_($index2) [new Application/Traffic/CBR]
        $cbr_($index2) set packetSize_ 512
        $cbr_($index2) set interval_ 4.0
        $cbr_($index2) set random_ 1
        $cbr_($index2) set maxpkts_ 10000
        $cbr_($index2) attach-agent $udp_($index2)
        $ns_ connect $udp_($index2) $null_($index2)
        $ns_ at $timed "$cbr_($index2) start"
        $ns_ at [expr $timed + 0.01] "$cbr_($index2) stop"
        $ns_ at $timed "$ns_ trace-annotate \"AP NODE $index2 of zone $temp forwards
request message to SP NODE $index \""
            set timed [expr $timed + 0.02]
            $node_($access_point) set sp_id($cn) $index
        $node_($access_point) set service_id($cn) $serv
        set cn [ expr $cn + 1 ]
        $node_($access_point) set mem_count $cn

      #SP sends hit message to SR
        set index2 "[ $node_($leader($temp)) set sp($temp1) ]"
        set index $sr
            set udp_($index2) [new Agent/UDP]
            $ns_ attach-agent $node_($index2) $udp_($index2)
            set null_($index2) [new Agent/Null]
        $ns_ attach-agent $node_($index) $null_($index2)
        set cbr_($index2) [new Application/Traffic/CBR]
        $cbr_($index2) set packetSize_ 512
        $cbr_($index2) set interval_ 4.0
        $cbr_($index2) set random_ 1
        $cbr_($index2) set maxpkts_ 10000
        $cbr_($index2) attach-agent $udp_($index2)
        $ns_ connect $udp_($index2) $null_($index2)
        $ns_ at $timed "$cbr_($index2) start"
        $ns_ at [expr $timed + 0.01] "$cbr_($index2) stop"
        $ns_ at $timed "$ns_ trace-annotate \"SP NODE $index2 sends hit message to SR
NODE $index \""
        set timed [expr $timed + 0.02]
        set c "[ $node_($index2) set sr_count ]"
          $node_($index2) set req_id($c) $index
          $node_($index2) set req_ap($c) "[ $node_($index) set lc ]"
          set c [ expr $c + 1]
          $node_($index2) set sr_count $c

         #SR responds to SP with an acknowledgement(delivery)
        set index2 $sr
        set index "[ $node_($leader($temp)) set sp($temp1) ]"
          set udp_($index2) [new Agent/UDP]
          $ns_ attach-agent $node_($index2) $udp_($index2)
          set null_($index2) [new Agent/Null]
        $ns_ attach-agent $node_($index) $null_($index2)
        set cbr_($index2) [new Application/Traffic/CBR]
        $cbr_($index2) set packetSize_ 512
        $cbr_($index2) set interval_ 4.0
        $cbr_($index2) set random_ 1
        $cbr_($index2) set maxpkts_ 10000
        $cbr_($index2) attach-agent $udp_($index2)
        $ns_ connect $udp_($index2) $null_($index2)
        $ns_ at $timed "$cbr_($index2) start"
        $ns_ at [expr $timed + 0.01] "$cbr_($index2) stop"
        $ns_ at $timed "$ns_ trace-annotate \"SR NODE $index2 sends response message
to SP NODE $index \""
      } else {
         $ns_ at $timed "$ns_ trace-annotate \"Service $serv not available.... Retry later....
\" "
      }}}}

# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
   $ns_ at 99.05 "$node_($i) reset";
}
$ns_ at 20.2  "zone_construct 20.2"
$ns_ at 20.2 " elect1 20.20 20.28 "
$ns_ at 20.2 " elect2 20.20 20.24 "
$ns_ at 20.2 " elect9 20.51 20.55 "
```

```
$ns_ at 31.2 "update_table_ap 31.2 31.3"
$ns_ at 31.2  "update_neighbor 31.2"
$ns_ at 31.2 "elect_rp 31.2"
$ns_ at 31.2 "update_table_rp  31.2"
$ns_ at 32.0 "ap_fail 32.0"
$ns_ at 32.9 "service_discovery 32.9 15 19"
$ns_ at 35.2 "add_sp 35.2"
$ns_ at 37.2 "del_sp 37.2"
$ns_ at 39.2 "new_node 39.2 40.2 "
$ns_ at 52.9 "service_discovery 52.9 9 1"
$ns_ at 95.9 "service_discovery 95.9 3 4"

for {set i 26 } {$i < 99 } {incr i} {
 $ns_ at $i  "zone_construct $i"
 $ns_ at $i "update_neighbor $i"
 $ns_ at $i " elect1 [ expr $i + 0.02]  [ expr $i + 0.04]  "
 $ns_ at $i " elect2 [ expr $i + 0.03]  [ expr $i + 0.05]  "
 set i [ expr $i + 15 ]
}

for {set k 32 } {$k < 99 } {incr k} {
$ns_ at $k  "hello_beacon $k [expr $k + 0.02]"
 set k [expr $k + 11]
}


for {set k 33 } {$k < 99 } {incr k} {
  $ns_ at $k " rp_move $k"
  set k [expr $k + 1]
}
$ns_ at 99.92 "stop"

$ns_ at 99.93 "puts \"NS EXITING...\" ; $ns_ halt"

proc stop {} {
   global ns_ tracefile1
   $ns_ flush-trace
   close $tracefile1
   puts "running nam..."
   exec nam 15-m-idbased &
   }

puts "Starting Simulation..."
$ns_ run
```

APPENDIX

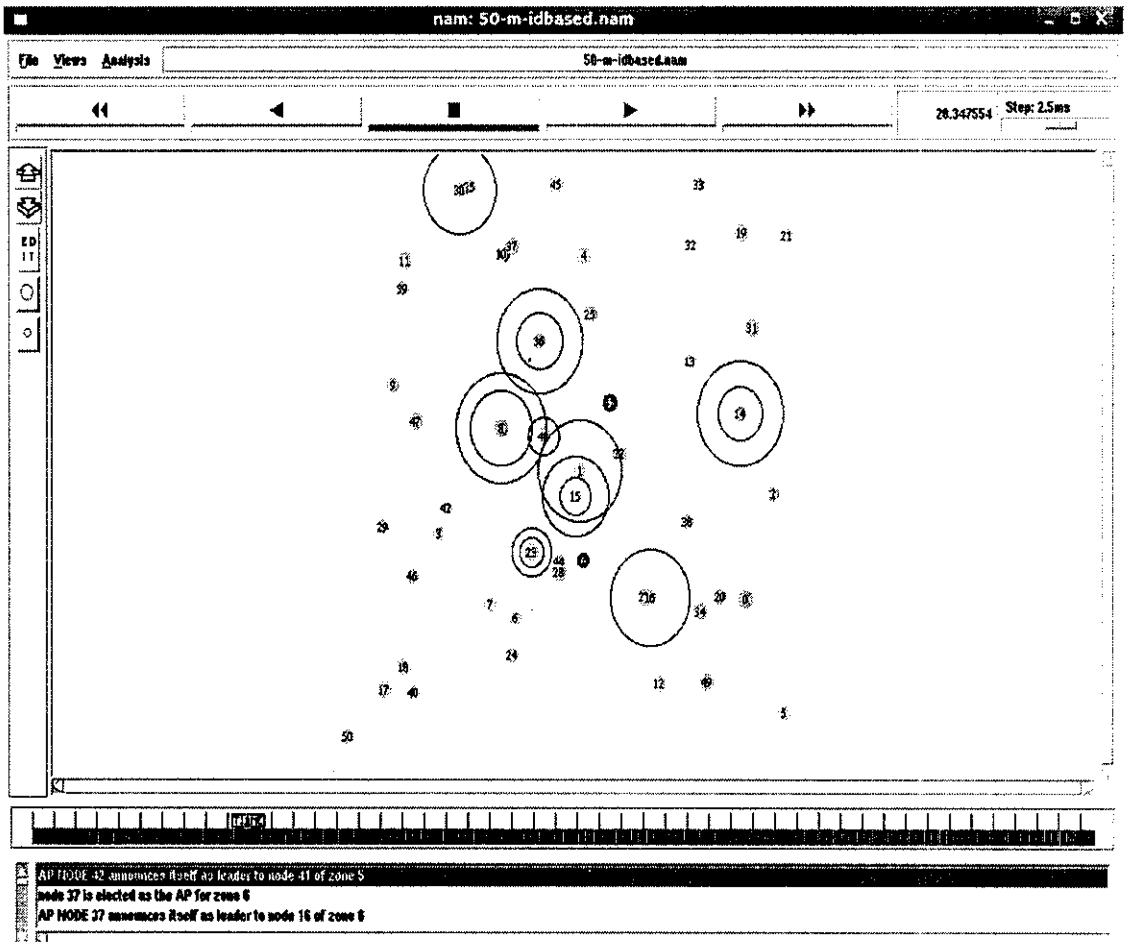
SCREENSHOTS

Service Registration



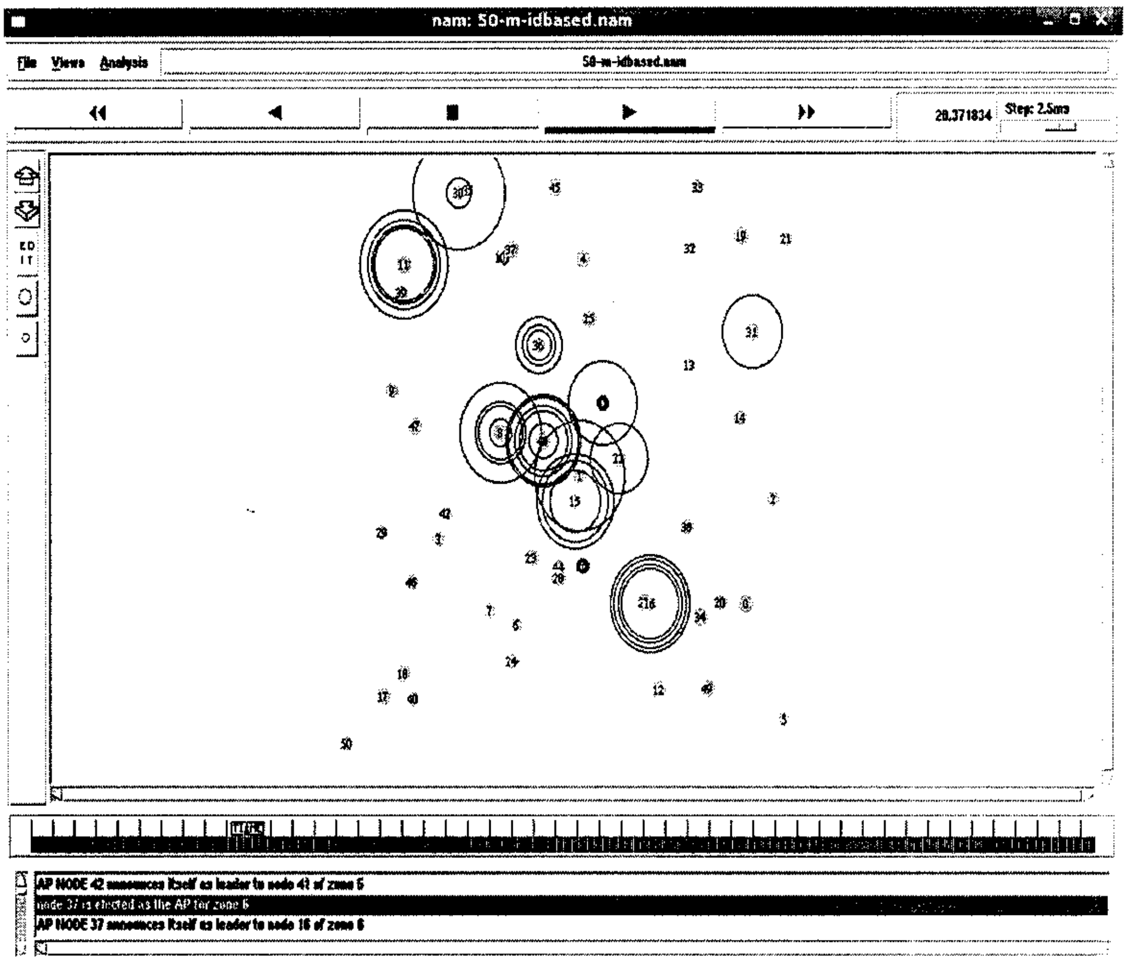
```
SP NODE 32 REGISTERS its SERVICES WITH AP NODE 35 of zone 0
SP NODE 48 REGISTERS its SERVICES WITH AP NODE 2 of zone 1
Neighbor 34 is updated in 0 's neighbor table
```

Service Announcement



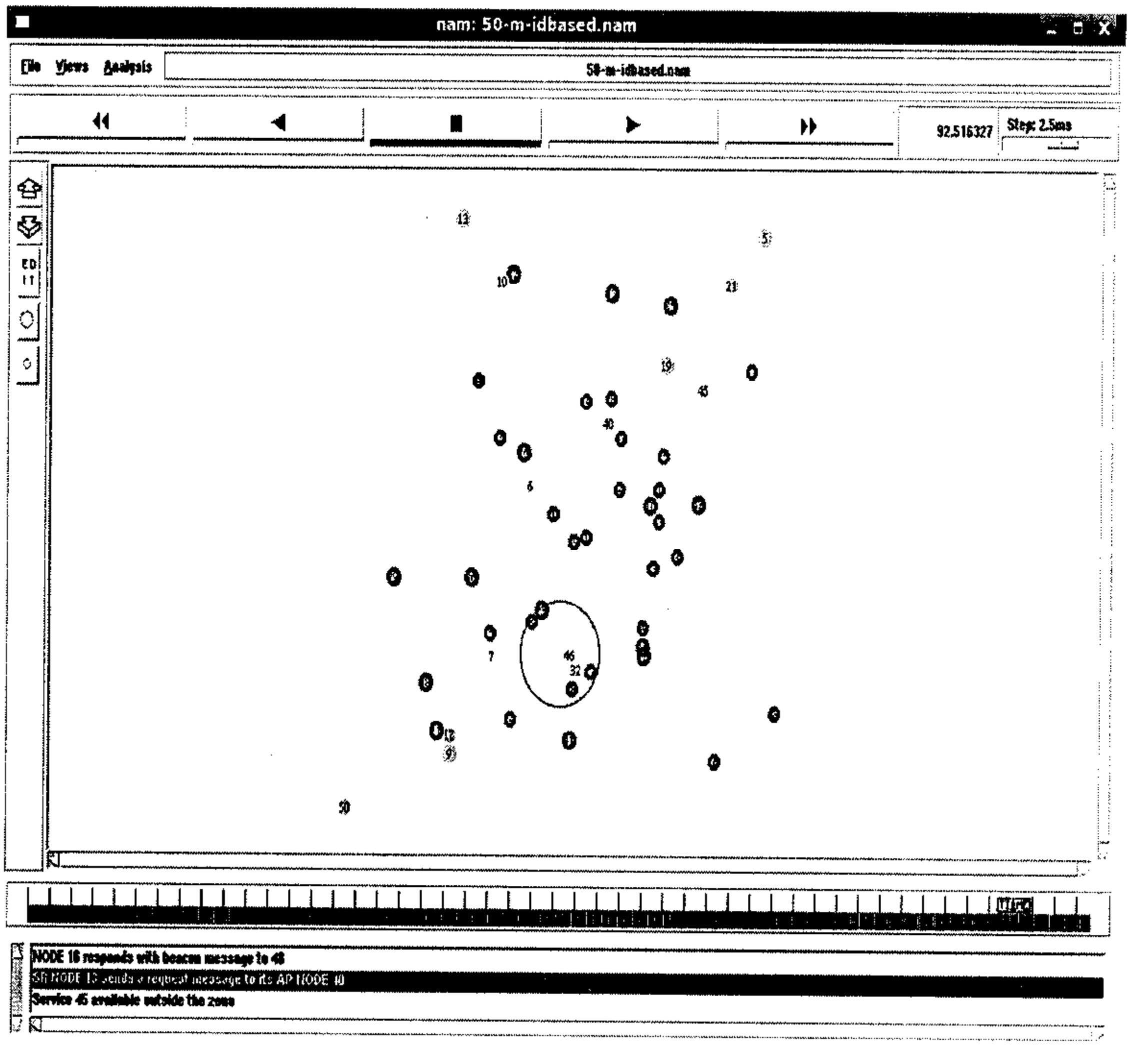
```
AP NODE 42 announces itself as leader to node 41 of zone 5
node 37 is elected as the AP for zone 6
AP NODE 37 announces itself as leader to node 16 of zone 6
```

Leader Election



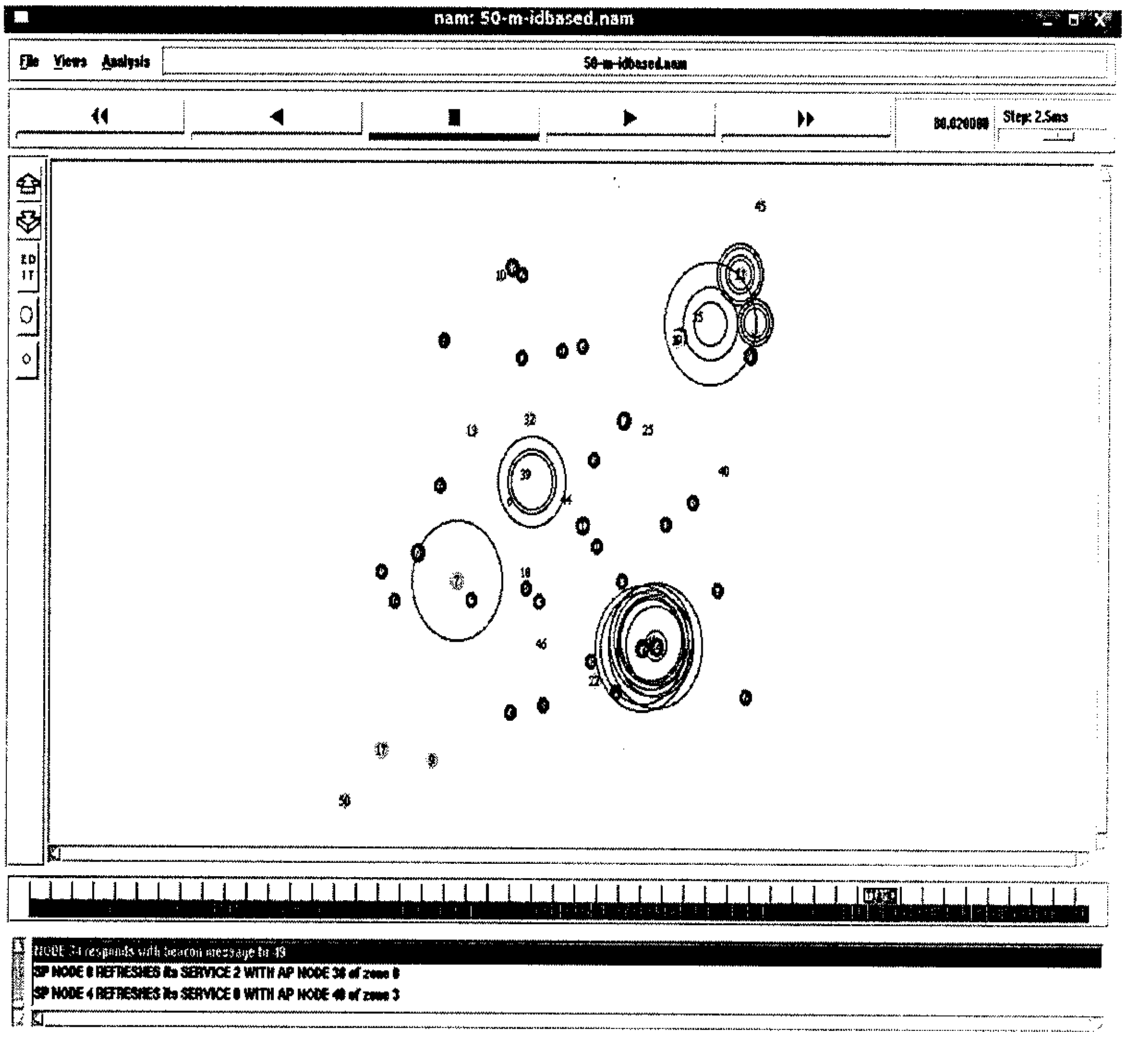
```
AP NODE 42 announces itself as leader to node 41 of zone 6
node 37 is elected as the AP for zone 6
AP NODE 37 announces itself as leader to node 16 of zone 6
```

Service request



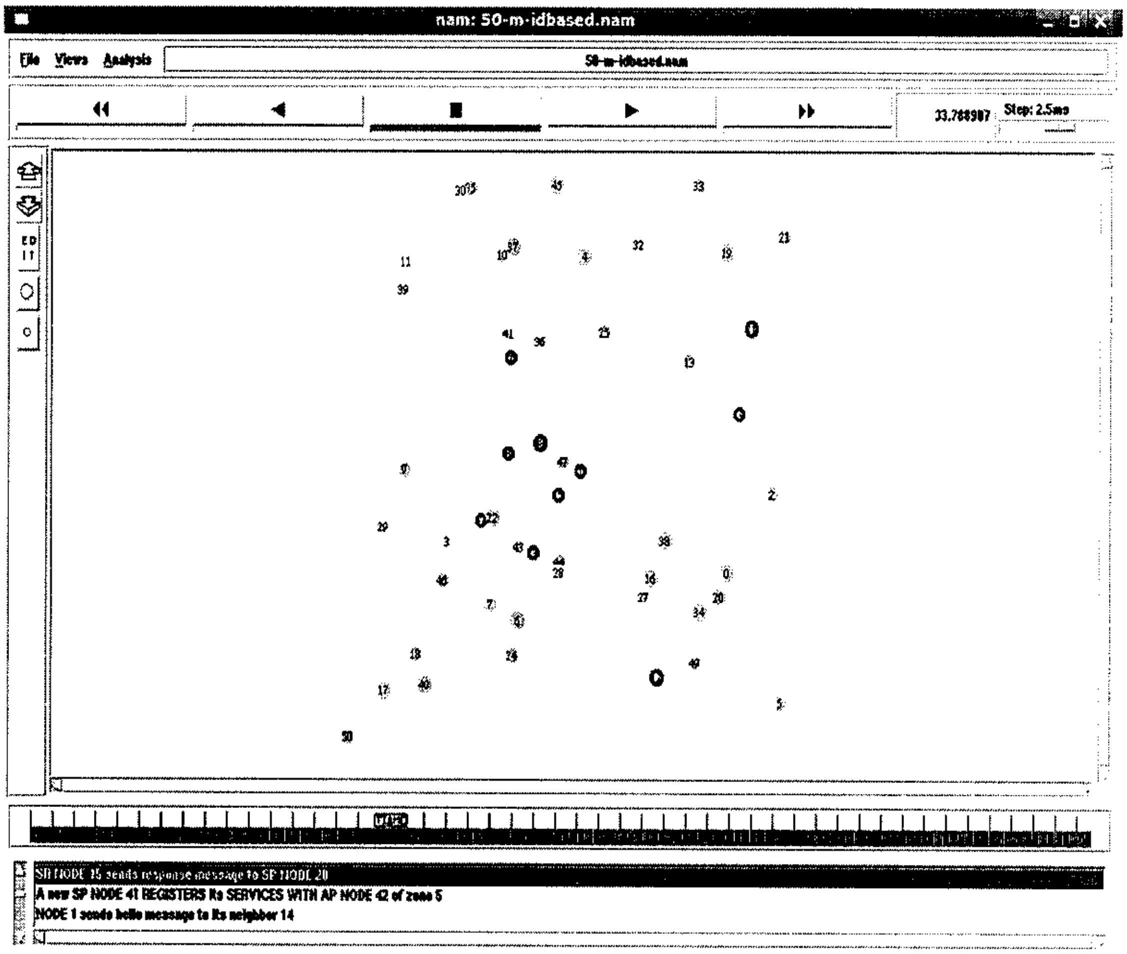
```
NODE 16 responds with beacon message to 48
SR NODE 13 sends a request message to its AP NODE 40
Service 45 available outside the zone
```

Service Coordination



```
NODE 34 responds with beacon message to 49
SP NODE 8 REFRESHES its SERVICE 2 WITH AP NODE 38 of zone 8
SP NODE 4 REFRESHES its SERVICE 8 WITH AP NODE 40 of zone 3
```

Service Delivery



```
SB NODE 15 sends response message to SP NODE 20
A new SP NODE 41 REGISTERS its SERVICES WITH AP NODE 42 of zone 5
NODE 1 sends hello message to its neighbor 14
```

REFERENCES

[1] Xiaojing Xiang and Xin Wang, "A Scalable Geographic Service provision framework for Mobile Adhoc Networks", in PerCom '07 IEEE,2007.

[2] Guo Song , Gerard Parr and Bryan Scotney, " A conceptual Framework for bandwidth Protection in Mobile Adhoc Networks", in MOBICOM IEEE,2006.

[3] Saumitra M.Das, Himabindu Pucha and Y.Charlie Hu, "Distributed Hashing for Scalable Multicast in Wireless Adhoc Networks", in WowMom'06 IEEE, 2006.

[4] Xiaojing Xiang and Xin Wang, "An Efficient Geographic Multicast Protocol for Mobile Adhoc Networks", in WoWMoM'06 IEEE, 2006.

[5] Dimitrios Koutsonikolas, Saumitra M.Das, Y.Charlie Hu, Ivan Stojmenovic, "Hierarchical geographic multicast routing for wireless sensor networks", in Wireless networks Springer, Oct 2008.

[6] Fei Li, Shile Zhang, Xin Wang, Xiangyang Xue, and Hong Shen, " Vote-based Clustering Algorithm in Mobile Ad-hoc Networks", in MANET, 2004.

[7] Jaspreet Kaur, Cheng Li, " Simulation and Analysis of Multicast protocols in Mobile Ad Hoc Networks using NS-2", in MANET.

[8] Ranwa Al Mallah and Alejandro Quintero, " A Light-Weight Service Discovery Protocol for Ad Hoc Networks", in Mobile Computing and Networking Research Laboratory, Journal of computer Science 5 (4):330-337, 2009.

[9] Shuhui Yang and Jie Wu, "New Technologies of Multicasting in MANET", in IEEE,2004.

[10] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks: performance evaluation and optimization", *Computer Networks*, 1:155–174, 1977.

[11] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks", IEEE *Trans. Mobile Computing*, 5(1), January 2006.