ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

PACL-SDP: An efficient Cross-Layered Service Discovery Architecture for Mobile Ad Hoc Networks

By

Dawit Abera

A thesis submitted to the school of Graduate studies of Addis Ababa University in partial fulfillment of the requirements for the degree of

Masters of Science in Computer Engineering

July 2010

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

PACL-SDP: An efficient Cross-Layered Service Discovery Architecture for Mobile Ad Hoc Networks

By

Dawit Abera

Advisor

Prof. Dr Sayed Nouh

Declaration

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been fully acknowledged.

Name: Dawit Abera Signature: _____

Place: Addis Ababa

Date of submission: July, 2010

This thesis has been submitted for examination with my approval as a university advisor.

Prof. Dr Sayed Nouh Signature: _____
Advisor's Name

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

PACL-SDP: An efficient Cross-Layered Service Discovery Architecture for Mobile Ad Hoc Networks

By

Dawit Abera

APPROVAL BY BOARD OF EXAMINERS

Dr. Mengesha Mamo	_____
Chairman Department of Graduate Committee	Signature
Prof. Dr. Eng. Sayed Nouh	_____
Advisor	Signature
Dr.-Ing. Hailu Ayele	_____
Internal Examiner	Signature
Dr. Mulugeta Libsie	_____
External Examiner	Signature

CONTENTS PAGE

Abstract

Efficient service discovery is a key for any successful network communication. The necessity is even emphasized for mobile ad hoc networks that do not rely on any infrastructure to maintain connectivity. The process of resource discovery and the corresponding service acquisition should be carried out with optimum consumption of energy and within a reasonable response time period to keep the life of the battery operated devices.

In this thesis work a new service discovery architecture is proposed following the guidelines of Service Location Protocol standard from IETF[†]. The new solution is a cross-layered protocol based on a distributed directory design that considers physical proximity of devices for service discovery. The new proposal has been compared with a traditional application layer based service discovery protocol with the same distributed directory architecture that was designed from the same standard, SLP. Both protocols were implemented in NS2 for comparison. The results of the findings are compelling in that the cross-layered approach yields a much better performance for various evaluation metrics used. Under the various scenarios considered, the new cross-layered protocol has an average energy saving ranging from 10.24 % to 13.73%. The observed packet delivery ratio shows an average increase from 3.49% to 5.76% for the recommended mobility models. The service availability has also increased on average from 8.14% to 67.47% for those mobility models. In addition, the average raise of throughput is observed in the range of 2.64% to 8.91%.

[†] Internet Engineering Task Force

Acknowledgment

I would like to thank my advisor Prof. Dr. Eng. Sayed Nouh for his patience, constructive advice, and valuable feedbacks during the course of the work.

I am also much indebted to Dr. Tracy Camp from Colorado School of mines for her willingness to send me several mobility model packages. Also, special thanks go to Dr. Mirco Musolesi form University College London for sharing his new community based mobility model available at a public domain.

Dawit Abera
May 3, 2010

List of Tables

Page

List of Figures Page

List of Acronyms

4G	Fourth Generation
ACK	Acknowledge
AMS	Agent Management System
AODV	Ad Hoc On-Demand Distance Vector
API	Application Programming Interface
AVERT	Adaptive SerVicE and RouTe discovery protocol
CBR	Constant Bit Rate
DA	Directory Agent
DC	Directory Cache
DF	Directory Facilitator
DNS	Domain Name Service
DHCP	Dynamic Host Configuration Protocol
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
eXML	Extensible Markup Language
FIFO	First In First Out
GDB	GNU DeBugger
GNU	GNU is Not Unix
HARP	Hybrid Ad hoc Routing Protocol
HTTP	Hyper Text Transfer Protocol
IBM	International Business Machine
IETF	Internet Engineering Task Force
IZR	Independent Zonal Routing
JINI	Java Intelligent Network Infrastructure
LAN	Local Area Network
LB	Lower is Better
LC	Local Cache
LS	Leader Selection
MAC	Media Access Control

MANET	Mobile Ad hoc NETwork
MPR	MultiPoint Relay
MSD	Mercurey Service Discovery
NC	Neighbours' Cache
NS-2	Network Simulator ver. 2
ODMRP	On-Demand Multicast Routing Protocol
OLSR	Optimized Link-State Routing
OSI	Open System Interconnection
PACL	Proximity Aware Cross-Layered
PDA	Personal Digital Assistant
QoS	Quality of Service
RLD	Routing Layer Driver
RERR	Route ERRor
RREP	Route REPly
RREQ	Route REQuest
RPGM	Reference Point Group Mobility
SA	Service Agent
SAP	Service Access Point
SANDMAN	Service AwareNess and Discovery for MANETs
SC	Service Coordinator
SDL	Service Discovery Library
SDP	Service Discovery Protocol
SLP	Service Location Protocol
SLP_MANET	Service Location Protocol for MANETs
SREP	Service REPly
SREQ	Service REQuest
SSD	Successful Service Discovery
TCP/IP	Transmission Control Protocol/Internet Protocol
UA	User Agent
UPnP	Universal Plug and Play

Wi-Fi	Wireless-Fidelity
WLAN	Wireless Local Area Network
WRP	Wireless Routing Protocol
WSN	Wireless Sensor Network
ZRP	Zonal Routing Protocol

Chapter One

Introduction

1.1 Background

Computing networks have long been the standard for resource sharing and means of communication. Nowadays, it is becoming increasingly rare to find sectors that operate without the aid of computing networks. It is true that the contemporary network technologies and architectures have grown to a high standard that today's networks are much more efficient and reliable than those decades before.

On the other hand, human needs to possess an even simpler and friendlier network environment keep on growing, making the advances in the computing industry non-stop. One area of the field where rapid interest is continuing to develop is the wireless networks. People are equipped with many kinds of small handheld devices that have made life much richer for them.

Wireless networks existing today may be broadly classified into two major classes, based on how the network is constructed and the underlying network architecture [22]. The first one is infrastructure-based wireless networks where the network nodes communicate through an access point. Among such types of networks are wireless local area networks (WLANs) with Wi-Fi hotspots, cellular networks, and satellite networks. The second class of wireless networks comprises networks with no infrastructure. Members of these networks can directly communicate with one another without the aid of a third component. Among these networks are Bluetooth Networks and mobile ad hoc networks. The focus of this thesis work is on mobile ad hoc networks.

A Mobile ad hoc network is a collection of wireless heterogenous mobile nodes (or routers) dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration [7]. Such a network is composed of mobile nodes like portable computers, personal digital assistants (PDAs), cell phones, etc. Each node can play three principal roles simultaneously within the network, namely a mobile node can be a client, a server, and a router.

Mobile ad hoc networks, abbreviated as MANETs, are spontaneously created autonomous networks. They are spontaneously created in the sense that they are self-configurable without any human intervention. They are autonomous in that they can exist by themselves, although it is possible to integrate them with external networks such as the Internet.

Mobile ad hoc networks are characterized by many features. The components of the network are dynamic and thus there are frequent topology changes. Most elements of the network are battery-operated. This implies that energy is a scarce resource in these networks. In addition, the available computational power is limited as the sizes of these devices get smaller.

Mobile ad hoc networks can find applicability in diverse areas. Among these are search and rescue operations, virtual classrooms, entertainment networks, military tasks, etc. Nowadays, the interaction of ad hoc networks with well-established networks such as the Internet is changing information access from “anytime anywhere” into “all the time, everywhere” [7] .

Hence, ad hoc networks find continued use in the following areas [7].

- Community network
- Enterprise network
- Home network
- Emergency network
- Vehicle network
- Sensor network

Ad hoc networking is expected to form the essential piece in the 4G (fourth generation) network architecture [22].

1.2 Problem Statement

Many mobile devices constituting an ad hoc network have limited resources. These devices cooperate among themselves to carry out tasks that can not be performed alone. This cooperation is usually manifested in terms of service consumers and service providers analogy. A very important consideration in such networks is how mobile nodes discover and locate services within a network in an efficient manner.

There have been several attempts to design service discovery architectures to meet the needs of mobile ad hoc networks [1]. Despite the efforts, most dwell on application layer based solution. As such, some of the basic problems associated with ad hoc networks such as optimum use of scarce energy and the reduction of overhead messages within the network are not well addressed. Consequently, the search for an effective service discovery protocol is still an open issue.

1.3 Motivation

In recent years the idea of designing cross-layered service discovery protocols is gaining a lot of attention. Indeed, ad hoc networks greatly benefit from such designs. For example by integrating the service discovery protocol with the underlying routing layer protocol, it is possible to exploit features available in the lower layer to transport service discovery messages without expending any extra energy for the service discovery action. This helps the mobile node in saving considerable amount of energy. On top of this, the response times the node experiences for service request-response processes could be substantially reduced. The motivation behind this thesis work arises from the attractiveness of cross-layered service discovery architectures.

1.4 Objective

1.4.1 General Objective

Investigating the impact of a new cross-layered Ad hoc On Demand Distance Vector (AODV) based service discovery protocol on the performance of an ad hoc mobile network with respect to energy utilization, efficiency of successful discovery, service discovery time, and communication message overhead.

1.4.2 Specific Objectives

- Investigating how a cluster based MANET service lookup system can play a role to facilitate service discovery by mobile hosts.
- Studying the effect of a routing-layer based service discovery protocol by modifying the AODV reactive protocol.
- Proposing a context aware, cross-layered service discovery protocol based on the above modified version of AODV that also incorporates one or more of the following properties; physical proximity, service lifetime, energy availability, etc

1.5 Methodology

The research work used the following methodologies.

Literature survey: An extensive literature reviews on the subjects of leader election algorithms and service discovery protocols in MANETs have been carried out.

Analysis and Design: Mechanisms for reducing network congestion during service discovery has been analyzed. Context awareness in terms of physical proximity has been used to enhance energy utilization and reduce delays incurred during service discovery. The findings have been implemented for testing and verification. The performance of the new algorithm has been assessed by simulation for mobile networks with different mobility models. Finally, a comparative analysis of the results obtained for the proposed solution with a traditional application layer service discovery protocol has been carried out.

Platform Used: The platform used for the research work possesses the following software and hardware components:

- Operating System: Open SuSE (ver. 10.3)
- Simulator: NS2 (ver. 2.34)
- Hardware: Dual-core processor machine 2.2 GHz, 996 MB memory

1.6 Contributions

There are two main contributions in this thesis work. The first one is a local leader election algorithm that optimizes the use of resources by mobile ad hoc networks while forming clusters. The algorithm elects a resource reach node that can serve as a service coordinator within its cluster. The algorithm is designed in such a way that it tries to minimize unnecessary broadcast messages while the leader election process is carried out as well as when leader state information is maintained within the network.

The elected leader will become the head of the cluster and members of the cluster interact with the cluster head either for service registration by servers, or for service inquiry by clients.

The second contribution is the design of a cross-layered service discovery protocol for the ad hoc networks. In this algorithm, the upper layer based service discovery protocol is integrated with the lower layer routing protocol to facilitate service advertisement and discovery processes. The cooperation between the layers is used to an advantage for efficient utilization of scarce energy, to minimize communication message overheads within the network, and reduce the amount of time needed for service discovery among other things.

1.7 Scope and Limitations of the thesis work

The discussions and implementation details presented in this work apply for a single cluster composed of a number of nodes. Inter-cluster aspects of the service discovey architecture are not part of the work. In addition, only a single leader for a given cluster has been assessed. The possibility of electing additional secondary or tertiary leaders as standby elements can be considered as a continuation of the work.

1.8 Thesis layout

The thesis work is organized as follows. The second chapter gives a brief discussion of mobile ad hoc networks focusing on unicast routing protocols. Chapter 3 dwells on existing leader election algorithms and service discovery architectures for mobile ad hoc networks. It describes the different approaches used in academia for proposing resource discovery protocols for ad hoc networks. Chapter 4 delivers an in-depth discussion on the proposed solution for the new cross-layered service discovery architecture. It explains the different algorithms and accompanying structures that have been studied to implement the protocol.

Chapter 5 discusses the implementation details of the new proposed solution in the NS2 software. The comparative analysis of the new cross-layered protocol with the traditional approach is presented for various network metrics under different mobility conditions. Finally, chapter 6 presents the conclusion that is drawn from the study and proposes some future work.

Chapter Two

Mobile ad hoc networks

The maturity of wireless transmissions and popularity of portable computing devices have enabled users to remain connected to networks on-the-fly. Such a network environment is called mobile computing or nomadic computing [28]. The majority of wireless networks found today are single-hop wireless networks where any two wireless hosts can reach one another only through a third network component. The third network element is a fixed network component which can be either a base station as in the case of cellular networks, or an access point as in the case of wireless LANs (WLANs).

There are several reasons why the wired backbone infrastructure may be unavailable for use by mobile hosts. For instance due to unexpected natural disasters. Also, it might be infeasible to construct sufficient fixed access points due to cost and performance considerations; for instance, having fixed network infrastructure in wilderness areas, festival grounds, or outdoor assemblies and activities is sometimes prohibitive. In addition for emergency search-and-rescue or military maneuvers, a quickly deployable temporary communication network may be needed.

In the aforementioned situations, a mobile ad hoc network (MANET) can be a better choice. A MANET consists of a set of mobile hosts operating without the aid of any established infrastructure of centralized administration such as base stations or access points. Communication is done through wireless links among mobile hosts through their antennas.

However, due to concerns such as radio power limitation and channel utilization, a mobile host may not be able to communicate directly with other hosts in a single hop fashion. In this case, a multi-hop scenario occurs, in which the packets sent by the source host must be relayed by several intermediate nodes. As such a node in an ad hoc environment plays three major roles. It can act as a server, a client, and as a router. The routing role of a node is vital for the existence of the network. On the other hand, it is also possible to create a network in which ad hoc networks can co-exist with the conventional fixed networks such as the Internet, forming what is known as a hybrid MANET, as shown below at Figure 2.1.

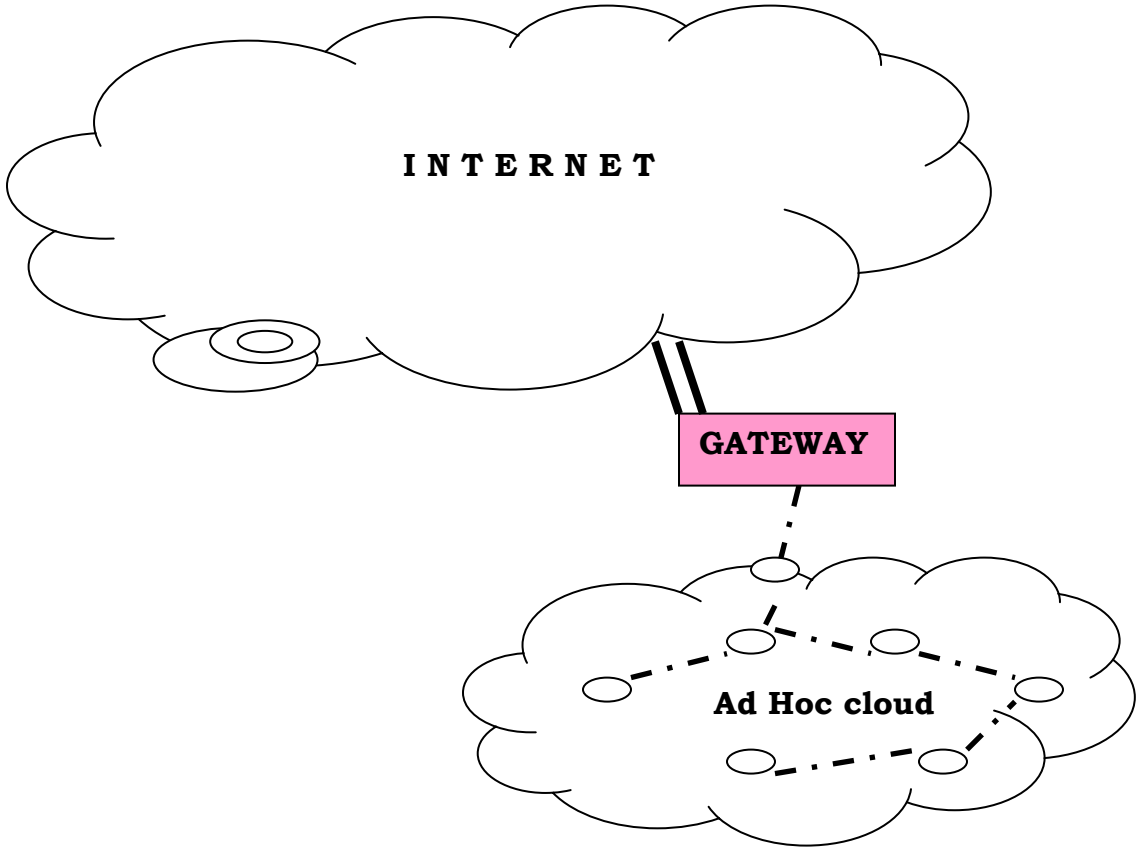

Fig. 2.1 A Hybrid MANET [4]

There are several factors that make mobile ad hoc networks different from conventional fixed networks or single-hop wireless networks. Mobile hosts are mostly battery operated. Thus, the life of the host in the network could be significantly reduced if the host is engaged in a highly processor intensive activities for prolonged period. Most of the nodes are handheld devices, and thus there may not even be enough processing capacity in terms of available memory and processor speed. In addition, owing to the mobile nature of the host, the radio coverage constantly varies. This results in frequent topology changes. When designing a routing protocol or any other network protocols for such systems, these and many other factors need to be taken into consideration. This makes the task quite challenging.

2.1 Unicast Routing Protocols for Ad Hoc networks

A number of routing protocols have been proposed for MANETs. One of the most popular methods of classifying routing protocols for ad hoc networks is based on how the routing information is acquired. Based on this feature there are three categories: reactive routing protocols, proactive routing protocols, and hybrid of the two [7].

*i) Reactive routing protocol*s

These are on-demand routing protocols in which information about the route to a destination is determined as needed by invoking a route discovery process. This could incur some delay in the communication but the strategy is vital for the life of the mobile host as it conserves considerable amount of energy. The most notable examples in this class of protocols are Ad hoc On- Demand Distance Vector (AODV) routing protocol, and Dynamic Source Routing (DSR) routing protocol.

ii) Proactive routing protocols

These are routing protocols that build routing tables regardless of the required route information. There is a periodic route update exchanges among nodes in the network to build updated routing table. Hence, upon a request of a route to a destination, the information is available beforehand without incurring any delay by processing a route discovery process as in the case of reactive protocols. However, this is at the expense of the scarcest resource of the host– the available energy. In addition, the frequent route update messages can consume considerable amount of bandwidth. Some examples of such type of routing protocols are Destination Sequenced Distance Vector (DSDV), and Wireless Routing Protocol (WRP).

iii) Hybrid routing protocols

This class of protocols combine the merits of the above two types and minimize the shortcomings of them. Usually these protocols employ a hierarchical routing protocol whereby the reactive and proactive features are implemented at different hierarchical levels. Some examples of such network protocols are Zonal Routing Protocol (ZRP), Hybrid Ad Hoc Routing (HARP) protocol.

In this thesis work the underlying routing layer protocol considered is AODV. A very brief overview of the protocol is discussed in the section to follow.

2.1.1 Ad Hoc On Demand Distance Vector routing protocol

The Ad hoc On-Demand Distance Vector (AODV) routing protocol is intended for use by mobile nodes in an ad hoc network. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within the ad hoc network.

AODV uses destination sequence numbers to ensure loop freedom at all times even in the face of inconsistent delivery of routing control messages, avoiding problems "Counting to infinity" associated with classical distance vector protocols [33].

The protocol is basically a combination of two protocols, DSDV (Destination-Sequenced Distance Vector) and DSR (Dynamic Source Routing) [12]. It adopted the on demand route discovery mechanism and route maintenance capability from DSR. It utilized features of hop-by-hop routing, sequence numbers and periodic beacons from DSDV.

One distinguishing feature of AODV is its use of a destination sequence number for each route entry. The destination sequence number is created by the destination for any route information it sends to requesting nodes. Using destination sequence numbers ensures loop freedom and is simple to program. Given the choice between two routes to a destination, a requesting node always selects the one with the greatest sequence number.

The AODV protocol is composed of four main control traffic types. The first one is a route request (RREQ), presented at Figure 2.2. It is broadcasted by a node that needs to connect to another node.

Source ID	Request ID	Source sequence No.	Destination Address	Destination sequence No.	Hop Count

Fig.2.2 packet format of a RREQ message

Whenever a fresh route to a destination is not available in the route cache of a source node, a RREQ is generated and flooded into the network. The request ID is incremented every time a new RREQ message is sent by the source. Hence, the combination of source address and request ID uniquely identifies the route inquiry packet.

The second type of control traffic is a route reply (RREP) message, shown at Figure 2.3, that is generated and unicasted to build the route towards the source of the RREQ message. It is possible to unicast a RREP message because every node that participated in the forwarding (broadcasting) of the RREQ message has built the reverse path towards the source of the RREQ message.

Source address	Destination address	Destination sequence No.	Hop count	Lifetime

Fig.2.3 Packet format of a RREP message

The third control traffic packet type is one that is used for establishing neighbouring connectivity by which nodes are aware of other nodes that are one hop away from each other. The messages from this control traffic are called HELLO messages. Finally, the route error (RERR) message is used to notify nodes whenever there are link breaks or losses of connectivity.

Figures 2.4 and 2.5 presented below show how RREQ and RREP messages are propagated in a typical ad hoc environment. Figure 2.4 shows the propagation of a RREQ message while Figure 2.5 shows the propagation of a RREP message. In the figures node 1 wants to communicate with node 8. After consulting its route cache, node 1 decides to broadcast a RREQ message to build a route towards node 8. Thus, node 1 is the source of the RREQ message and node 8 is the required destination node. When node 8 receives the RREQ message, it uses the node from which it received the first RREQ message as its next hop to reach node 1. Each intermediate node follows the same scheme to choose its next hop towards the source node, node 1. This way the RREP message is unicasted towards the source node thereby building the route between the source and destination nodes.

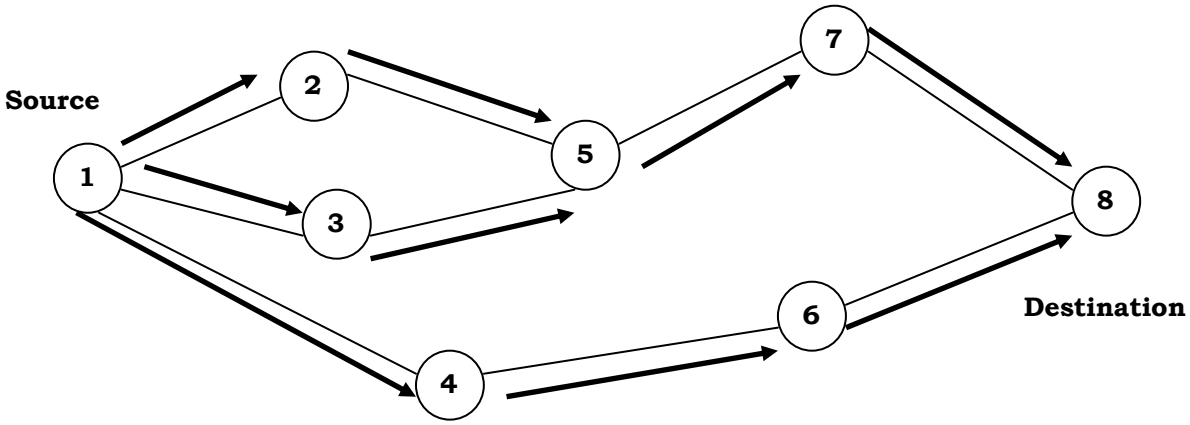

Fig. 2.4 Propagation of RREQ in AODV [12]

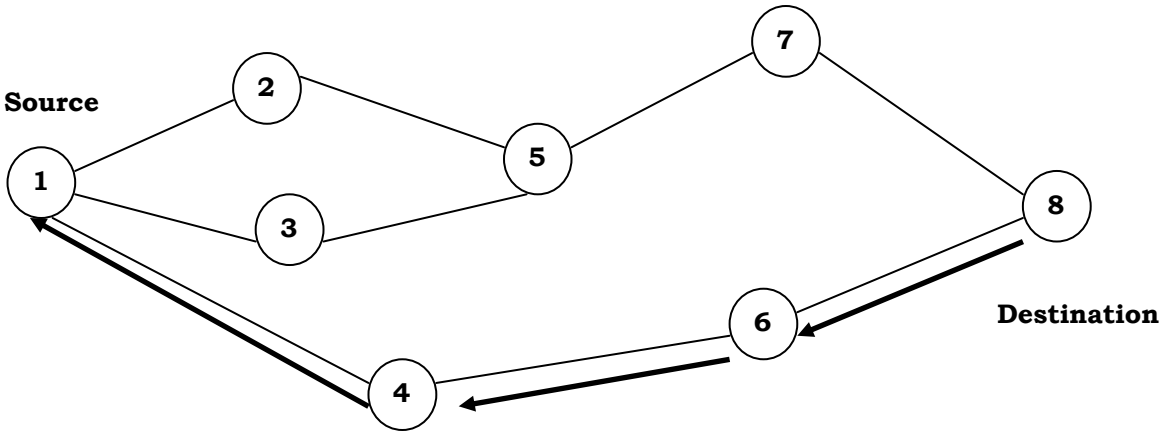

Fig. 2.5 Propagation of RREP in AODV [12]

2.2 Multicast Protocols for MANETs

Multicast protocols can be either source-based or core-based protocols based on how the multicast tree is constructed. In source-based protocols the multicast tree is constructed with reference to a source node and spans towards every other member node. Hence, there can be several multicast trees depending on the number of source hosts. In a core-based there is a single host serving as the reference to build the multicast tree [28]. Notable examples of multicast protocols for MANETs are On Demand Multicast Routing Protocol (ODMRP), and MAODV (Multicast AODV) which is an extension of the unicast AODV protocol. ODMRP falls into the first class of multicast protocols while MAODV belongs to the second category.

Chapter Three

Related Work

3.1 Leader election algorithms

Network nodes can elect a leader among them to serve different purposes. A leader carries out special duties such as:-

- Routing coordination
- Sensor coordination
- Directory services
- Key distribution

There have been several leader election algorithms in the research community developed or adopted for ad hoc networks [17, 18, 19]. Many of these works focus on designing algorithms that attempt to select leader nodes that are widely dispersed throughout the network. In these designs there is the inherent assumption that all nodes are capable of carrying out a leader's responsibility.

On the other hand, there are also a number of leader election algorithms that have been suggested for these networks based on election algorithms that use specific criteria to elect a leader. Most of these works are based on adaptations made for ad hoc networks from the classical termination detection algorithm for diffusion computations by Dijkstra and Scholten [21].

The work in [17] has used the growing and shrinking spanning tree technique. However, due to existence of frequent link failures in ad hoc networks, nodes may not receive acknowledgments from their neighbours. To avoid this problem they introduced "probe" and "reply" messages that keep neighbouring nodes to actively exchange messages throughout the election process. This step is not considerate to optimum use of energy. Moreover, the frequent message exchanges consume available bandwidth.

The study in [18] only extended the earlier work to include few more candidates to be used as leaders instead of electing only one leader. Otherwise, the frequent message exchanges are kept. Hence, this too suffers from the same problems.

The researchers in [19] designed a top-k leader election algorithm. Here too more than one leader is elected. Unlike the previous works, here the diffusion computation algorithm can be initiated by two or more nodes. Through consistent message exchanges these nodes, referred to as RED nodes, eventually elect top-k leaders among themselves. The problem of frequent message exchanges also applies to this method.

3.2 Service discovery processes

A service is any tangible or intangible thing or facility that can be used by any user, program, or another service [1]. Services can be available in either hardware or software types. Within a typical networking environment one can refer to services such as storage areas, print services, databases, etc.

In order to get benefit from services, a device should be capable of advertising its services, discovering other services, locating their providers, and be able to invoke the desired services. To carryout these activities, devices should be equipped with service discovery protocols [5].

Depending on the network type, service discovery protocols could take several forms. As different networks are characterized by differing features, a discovery protocol that is suitable for one type of network may not be useful for another type. As far as the research on service discovery protocols is concerned, there are three types of networks [1]. These are the wired networks, single hop wireless networks, and multi-hop wireless mobile ad hoc networks.

There have been several well established service discovery protocols in today's wired networks. Among the industry standard protocols for these static networks are Jini (by Sun Microsystems) [26], UPnP (by Microsoft) [27], SLP (by IETF) [29], Bonjour (by Apple) [25], and Salutation (by IBM) [24].

A very brief overview of some of these protocols is outlined below.

a) Jini

Jini is the name of a distributed computing system that can provide a spontaneous network plug and work. It is a middleware architecture based on the java programming language. Jini is a federation of clients and services that can be hardware, software, or a combination of both. In such an environment, services can announce their availability by registering at registry components. Clients can request for services by inquiring the registry components. The process takes place transparently with new services joining the network or old services leaving the network all without the intervention of the IT department.

b) UPnP.

Universal Plug and Play defines architecture for pervasive peer-to-peer interconnectivity of intelligent systems such as wireless devices, PCs of all form factors, home appliances. The UPnP architecture is independent of any programming language or platform. It is designed to support zero configuration, invisible networking, and automatic discovery of a wide variety of services.
The technologies leveraged in the architecture include Internet protocols such as IP, TCP, UDP, HTTP, and XML.

c) SLP

The Service Location Protocol (SLP) is a protocol meant for the discovery and use of network resources such as printers, web servers, fax machines, video cameras, file systems, backup devices (tape drives), databases, directories, mail servers, calendars, and many other types of services.

Accessing network services by manual configuration or administrative procedures will no longer be economically attractive. In the networked world of the future, interchangeable services will appear and disappear, and providing for the dynamic nature of their availability is an important accomplishment for SLP.

A further discussion of the architecture of SLP is important as it has been the basis for most of the discovery protocols commonly used in multiplatform implementations in commercial and non-commercial applications [6]. Because SLP is an IETF standard it is a protocol of choice for majority of researchers and users. Moreover, it is a stand-alone light-weight protocol that does not depend on any suit of protocols for its operation.

The architecture of SLP can be composed of three entities as depicted in Figure 3.1. These are user agents (UAs), directory agents (DAs), and service agents (SAs). User agents request services on behalf of users or applications. A service agent represents a service and carries out activities of periodical service advertisement to the whole network, for the case of passive service discovery, or limits its service registration to directory agents in directory based architectures, for cases of active service discovery. The service agent also responds to service queries from user agents. The third entity, the directory agent, is introduced to provide scalability to the protocol [29]. It is the agent that works on behalf of the directory node. Servers register services by communicating with the directory agent, and clients can query for services from directory agents. Network entities learn about the presence of directory agents in one of two ways. In the first method, DAs may periodically broadcast advertisements containing their unicast addresses and other directory related information to be used by clients and servers. In the second case, UAs and SAs may issue a multicast request looking for DAs, where DAs respond by sending directory advertisments, DAAdvert [29].

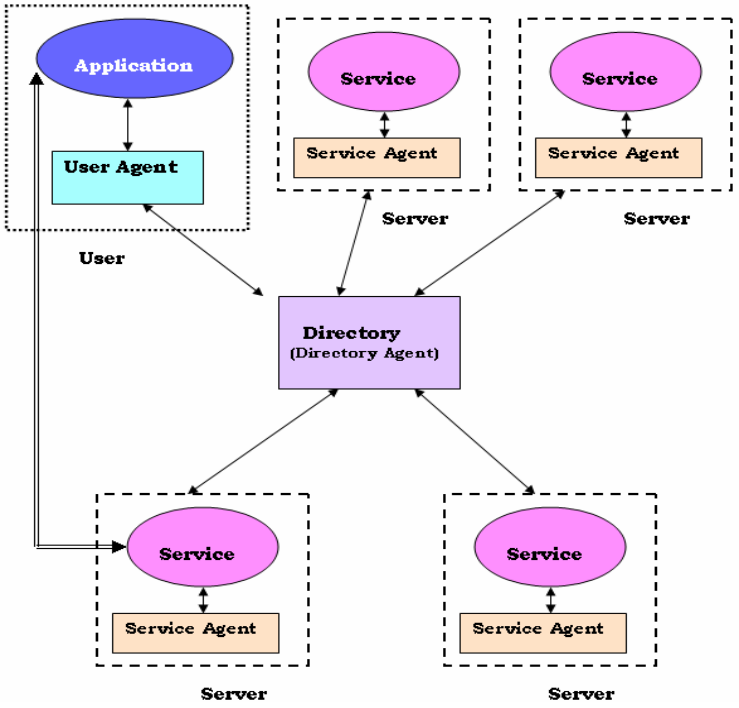

Fig.3.1 Directory based architecture of SLP [4]

3.3 Service Discovery protocols (SDPs) for MANETs

There are several challenges faced by researchers to design service discovery protocols (or any type of network protocol) for mobile ad hoc networks. These challenges stem from the inherent behaviours of ad hoc networks. Among these mentioned in [5] are:-

- Nodes are highly mobile in ad hoc networks
- Frequent topology changes or disconnections due to the dynamism of the environment
- Channel variability leading to significant communication variability such as data rate, delay etc.
- Scarcely available computational resources such as energy, memory capacity, and processor speed.

It is clear from the above characteristics of ad hoc networks that any network protocol for mobile nodes needs to be lightweight, i.e. requiring less processing and less storage requirement by the host. This ensures the longevity of the device in the network by conserving its battery powered energy.

The service discovery protocols outlined in section 3.1 can not be directly applied for ad hoc environment. They are designed with the assumption of devices with moderate processing capability, memory capacity, and ample source of energy.

In general, a service discovery process depends on the architecture of the underlying protocol. The architecture refers to the layout of any structure and how its major components are interconnected with one another [1].

A service discovery architecture mainly depends on the presence or absence of a directory. Based on the type of architecture, service discovery protocols for mobile ad hoc networks can be broadly classified into the same three categories.

- Service coordinator-based (Directory-based)
- Distributed (Directory-less)
- Hybrid of the two.

The directory based architecture may be implemented as centralized, hosted by a single host, or with distributed directories among different nodes. The former is common in wired networks. However, this approach is usually not suited to ad hoc environments due to mobility. The second approach is the use of multiple directory nodes serving different sections or groups called clusters within the network. This is referred to as the distributed directories architecture. In addition, it is possible to introduce secondary and tertiary directory nodes to serve as fallbacks within each group to enhance the availability of broker nodes for cases where the primary node is unreachable.

These properties of the distributed directories architecture make the approach ideally suitable for attaining good network performance and easily scalable topology for large networks. The authors in [1] have expressed their feeling on the presence of strong potential in directory-based with overlay support architecture for having scalable practicable real implementation of SDPs in MANETs.

Distributed directories are more suitable for MANETs. Hence, the term directory based SDP architecture for mobile ad hoc networks more often than not refers to the distributed directories implementation [5].

The schematic below, Figure 3.2, excerpted from [5] outlines the major classification types of service discovery architectures. Most of the service discovery protocols for mobile ad hoc networks are variants of the original SLP from Internet Engineering Task Force (IETF) with or without some optional features.

In the directory-based protocol, directory nodes advertise their presence in the network by periodically broadcasting their identity. Service providers register their services to these nodes. The type of information recorded at the registry nodes contains, among other things, the service provider address, the type of service offered, and service lifetime. Such an association between the address of a server and its offered service is referred to as a service binding.

The registration is usually accompanied by regular update. When clients require a service, they contact the directory nodes to learn the whereabouts of services required. The directory node responds by unicasting service binding information about the requested services.

The directory-less architecture can have two forms. The discovery process can be either server-initiated (push based) or client-initiated (pull based). In the first case servers periodically advertise their services by flooding the network with broadcast messages containing services they offer.

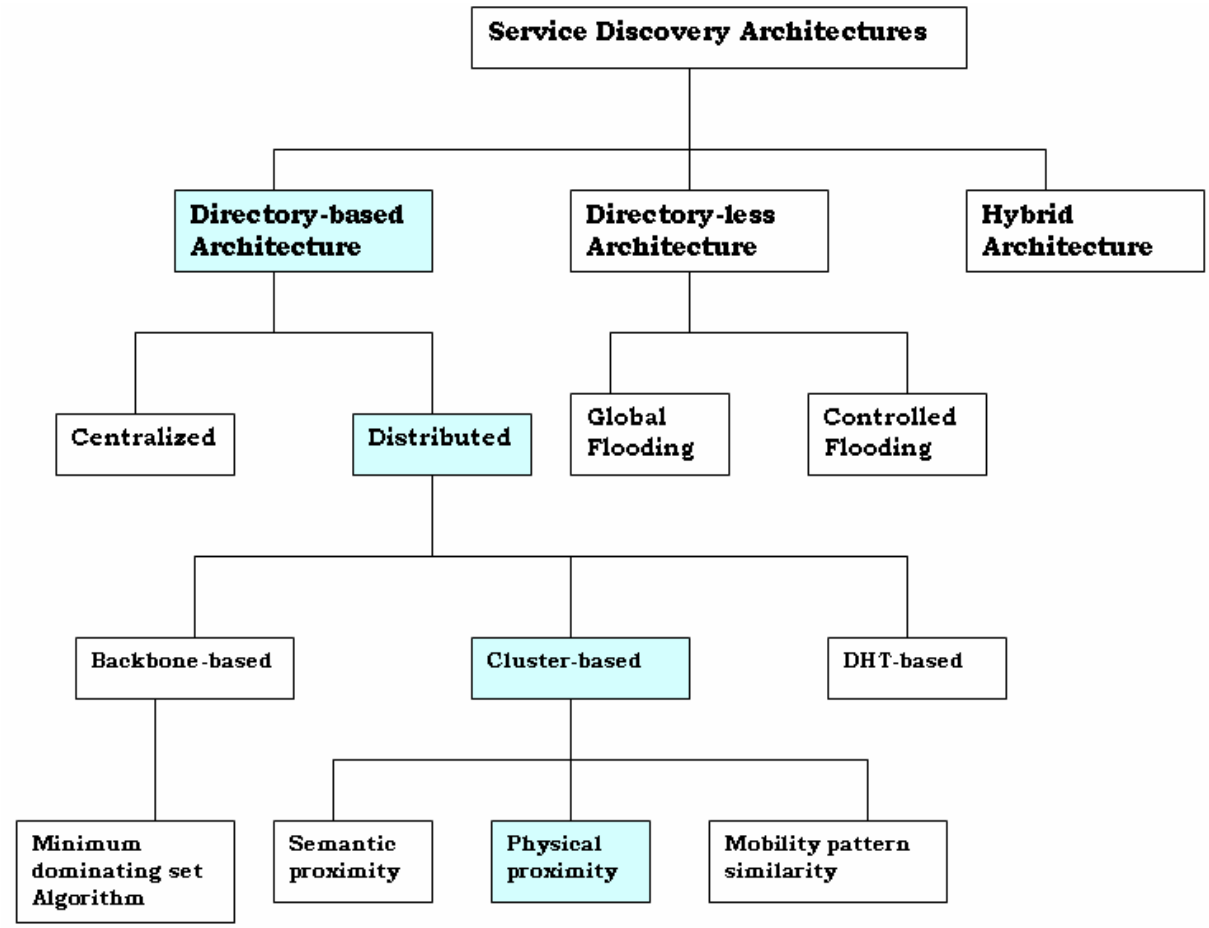

Fig.3.2 Major Service discovery architecture types [5]

Client nodes passively listen to such announcements. In the second case, clients requiring services broadcast their queries into the network. This is what is referred to as active discovery. Servers that can provide the requested services respond by unicasting their offer. In either cases, clients can choose among the servers based on certain criteria (such as hop distance, service lifetime, bandwidth, current load condition, etc) to invoke the service.

The hybrid architecture combines the two architectures. It creates a zone area within which the directory-less service discovery protocol is used. Outside the zone it employs the directory-based architecture. According to [1] hybrid architecture is mostly found in wired networks.

Another classification of service discovery protocols for MANETs is given in [1]. On the basis of the survey, SDP architectures are categorized into four classes:-

i) Directory-based with overlay support architecture
ii) Directory-based without overlay support architecture
iii) Directory-less with overlay support architecture
iv) Directory-less without overlay support architecture

An overlay network can be explained as follows as quoted in the paper. If a node that is part of an ad hoc network knows the address of another node within the same network and can communicate with it, then it is said that there is an overlay link between these two nodes.

An overlay link does not necessitate the existence of a direct physical or wireless link between the two nodes. The two nodes can form an overlay link while still they can reach each other through many intermediate nodes.

An overlay network is thus the collection of such overlay links and the nodes they connect. The resulting overlay network can be a structured one if there is some kind of organization among the nodes forming the network or it can be unstructured if no organization exists. In either case, there is a bootstrapping phase by which the overlay network is formed.

It is worth mentioning that whenever there is an overlay support within a network, it is a structured overlay that is referred to [1]. This is because the structured overlay network has the advantage of controlling the multicasts for service advertisement or query messages. In the next sections a short overview of SDPs from different perspectives is given.

a) Service Rings (Directory-based SDP with overlay support)

A service ring ideally groups together devices that are both physically close together and offer similar services [37]. Each ring possesses a designated service access point (SAP) which knows the summary of all services in its ring, as depicted in Figure 3.3. The service access points themselves can be connected through rings and thus forming a hierarchical structure of several orders. Once such an organization has been constructed, search messages can be localized to SAPs. A message descends into a sub ring only if its service access point knows such a service is delivered in its sub ring. This way the network is saved from unnecessary flooding of messages.

A Service ring is a distributed directories based approach for storing service information at different locations within the network. However, how the directories at the various locations are selected is not mentioned.

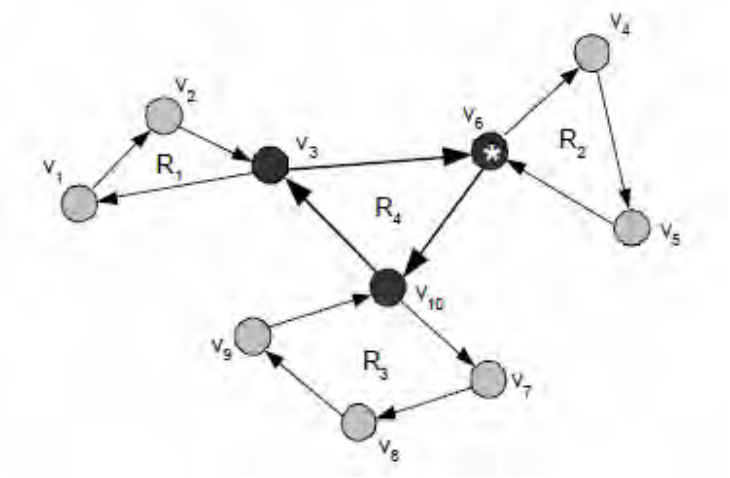

Fig. 3.3 Valid service ring overlay [37]

Arrows are representing overlay links. v1, v2, and v3 are forming the Level 0 ring R1 with v3 as its service access point (depicted as dark grey node). Also R2 and R3 are Level 0 rings. All their SAPs v3, v6, and v10 are forming the Level 1 ring R4, having the SAP v6 (depicted by the white asterisk). As there is only one world ring (R4), this overlay is a valid overlay.

b) Splendor (Directory-based without overlay support)

In splendor service discovery model there are four components. These are clients, servers, directories, and proxies [36]. The purpose of the proxies is to provide privacy for the service providers by enabling them to carryout authentication and authorization tasks. Directories periodically announce their unicast addresses by multicasting.

The information contained includes certificates, unicast network address and signatures on the address. In addition there are solicitation messages asking servers to register their services with them. Clients verify certificates before service lookups. During bootstrapping multicast addresses are used for initial communication among servers, clients, and directories.

All participating entities are required to possess a priori knowledge of these multicast addresses during bootstrapping. After bootstrapping all communications among the entities are carried through unicast messages.

Proxies serve as trusted service providers. They authenticate severs, manage keys for security, and carryout registration on their behalf. In general, the model is characterized by significant overhead for security.

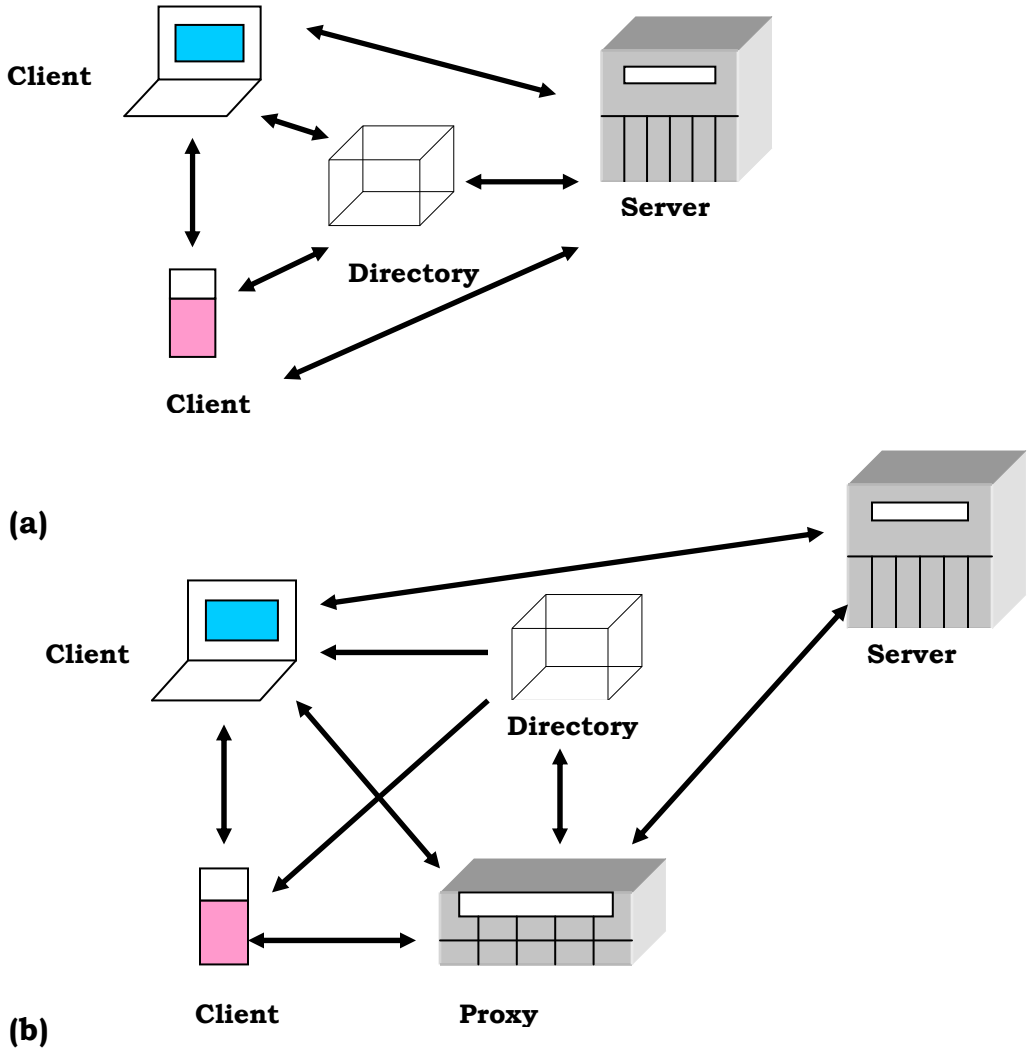

Fig. 3.4 Splendor service discovery models [36]

Two secure service discovery models with third-party servers are shown in Figure 3.4. (a) Client-service-directory model. (b). Client-service-directory-proxy model

c) Allia (Directory-less with overlay support)

Allia is a peer-to-peer services caching based architecture in which a node's participation in the ad hoc network is governed by a policy that is set by the user preferences [34]. To that end one of the components of the architecture is a policy manager module.

The operation of allia (alliance based service discovery) can be briefly explained as follows. Every node advertises its services to other nodes in its vicinity according to a local policy. A node receiving an advertisement may cache or reject the message based on its policy.

An alliance of a node is a set of nodes whose advertisements are cached by this node. Hence, a node knows members of its alliance, but doesn't know alliances where it belongs. Whenever a node moves, it forms new alliances by listening to new advertisements, and becomes member of other alliances by advertising its services.

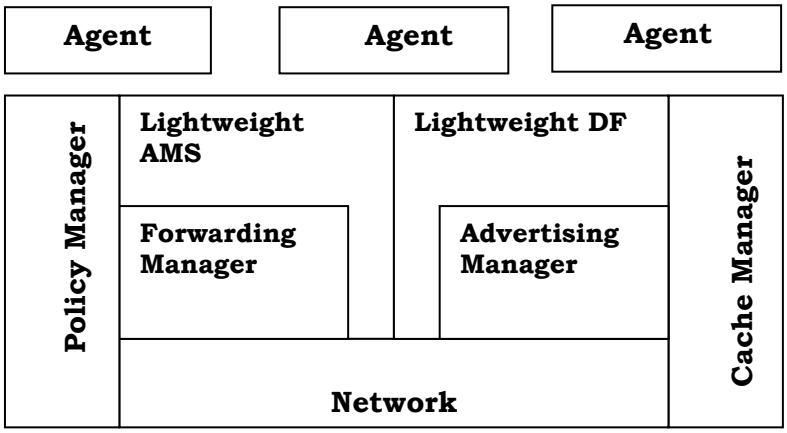

Fig. 3.5 Alliance Device Architecture [34]

Figure 3.5 shows the essential components of this architecture. The Policy Manager is responsible for enforcing policies to control platform behavior. The Cache Manager handles service advertisements from neighboring devices. The Advertising Manager actively broadcasts service descriptions registered to the local DF (Directory facilitator). The Forwarding Manager receives Service Advertisements and Request for Service messages. The DF registers service advertisements on the local system. The AMS (Agent management system) registers the local agents which are representatives of the available services on the host.

d) Konark (A directory-less architecture with no overlay support)

Konark is a service discovery and delivery protocol designed for ad hoc peer-to-peer networks. It is a middleware designed specifically for discovery and delivery of device independent services in these networks [35]. Konark is based on a model with each participating device having the capabilities to host its local services, deliver its own services using a resident micro-HTTP server for cross-platform service delivery, query the network for available services offered by others, and use the services it discovered in the network.

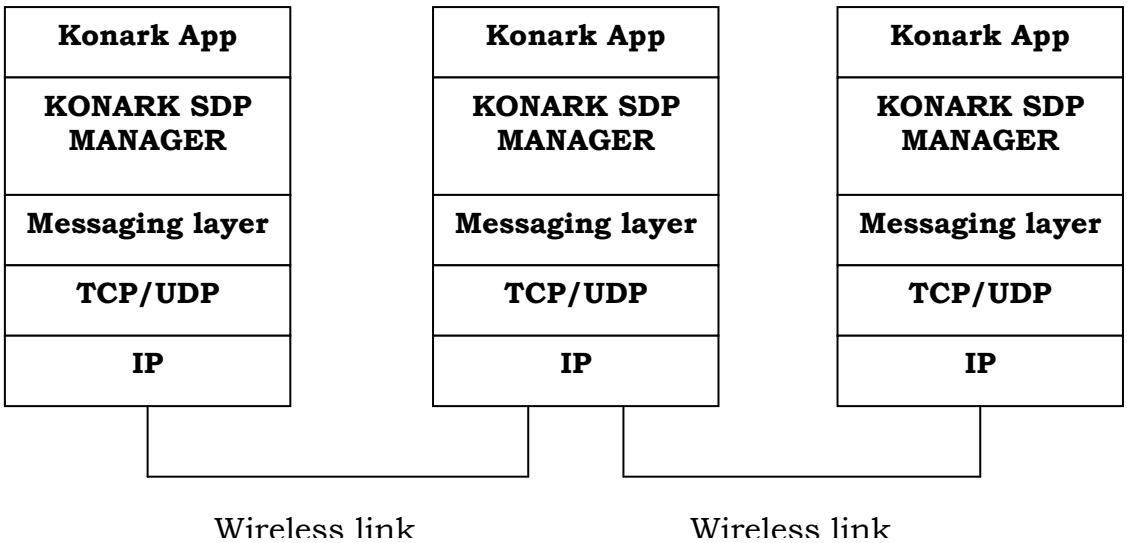

Fig. 3.6 Konark service discovery stack [35]

Figure 3.6 depicts the service discover protocol stack found in Konark. Each device includes a Konark Application that facilitates human interaction to initiate and control advertising, discovery, and service usage. It also includes SDP Managers and Registry that together maintain service objects and information about services offered by peers in the network. A micro-HTTP server is present to handle service delivery requests. Konark's two main parts - service discovery and service delivery – are constituted by these components.

f) SANDMAN: An energy efficient middleware for pervasive computing

SANDMAN (Service Awareness aND Discovery for Mobile Ad hoc Networks) is a directory based service discovery architecture that is conceived for energy conservation [2]. The motivation was the amount of energy expended by network interfaces of idle devices is high. In the research the authors came up with two operational modes for the mobile devices. These are a fully operational AWAKE mode, and an energy efficient SLEEP mode. In the SLEEP mode a device can not perform any operations or communications. Devices in SLEEP mode come into the AWAKE mode using internal timer. The criterion to set a device to a SLEEP mode was adopted from dynamic power management systems. Whenever the device is not indulged in any activity for predetermined idle threshold period, it is put into SLEEP mode. Figure 3.7 presents how the approach is organized.

Cluster nodes are always fully AWAKE. A node that goes into SLEEP mode notifies its cluster head the sleep period. The cluster head can modify the period however. It then sends back an ACK to the sleeping node. The sleeping node recalculates its sleep period and goes into sleep.

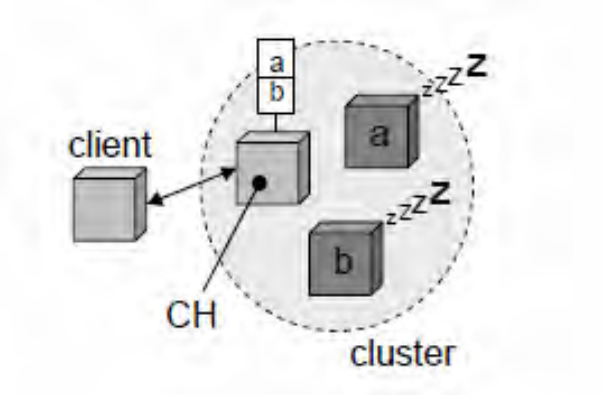

Fig. 3.7 SANDMAN approach [2]

If another node requests for a service to the cluster head and if this service is provided by a node in SLEEP mode, the cluster head sends the requesting device the period after which the sleeping nodes wake up. The requesting device has to wait for that remaining period for the server to wake up before contacting it. This is a downside of the architecture. In addition, if a sleeping node does not carryout any operation, then routes that could be discovered through this node are also affected.

3.4 Cross-layered Service discovery architectures for MANETs

The various types of discovery protocols for ad hoc networks presented thus far are all application layer based protocols. That is to say, the protocols are designed following the traditional layered discipline similar to the Open System Interconnection (OSI) reference model or the TCP/IP network model. However, in recent years a cross-layered approach has gained a lot of attention in the research fields as the benefits of the new approach by far exceed its demerits.

Nowadays, cross-layered network protocols are being applied in several fields of networking such as adaptive multimedia transmission in wireless networks [42], routing and security of multi-hop wireless networks [40], end to end QoS provision for wireless ad hoc networks [39], cooperative cross-layer design for wireless networks [41], power aware routing protocols [38], and also sensor networks [30].

In the next sub sections, some service discovery protocols for ad hoc networks that are implemented with cross-layer approach shall be discussed.

3.4.1 Extended Service Discovery by AODV (AODV_SD)

The first idea of extending on demand routing layer protocols to support service discovery was proposed by Koodli and Perkins in [31]. The already existing routing layer message formats may be extended to carry service-oriented information.

In [13] the researchers attempted to implement the idea by modifying the message formats for the AODV protocol. The RREQ (route request) and RREP (route reply) message formats have been extended to incorporate service information and are referred to as SREQ (service request) and SREP (service reply) respectively. Nodes acquire services when they participate in route or service discovery processes. The SREQ and SREP messages are processed following the same procedures similar to RREQ and RREP messages.

However, the proposal is solely based on the route discovery procedure which is a flooding process using the expanding ring search. As such, there is a potential for large number of frequent service query broadcasts within the network. This can significantly affect the performance of the whole network.

The problem could be aggravated as more and more clients request for several different services that are several hops away. Hence, the schema does not scale well for large number of nodes. Moreover, the scheme is not very attractive for memory conservation. Nodes with already limited amount of memory could be forced to exhaust the available space by caching several service advertisements. This would prevent the nodes from working on any other useful task due to lack of enough memory space.

3.4.2 A Cross-Layer Approach to Service Discovery and Selection

The service discovery architecture proposed in [20] consists of two major components. These are the service discovery library (SDL) which is independent of any routing layer protocol, and the routing layer driver (RLD) which is closely coupled with the routing layer. SDL isolates applications from the intricacies of the lower layer. It is the role of RLD to work with the routing layer protocols. Clients and servers interact with SDL to request service discovery messages and propagate advertisement messages.

Comparative analysis of the architecture with SLP for MANET for directory based and distributed architectures was made. The cross-layered architecture used DSR and DSDV as the routing layer protocol. However, directory agents were pre-loaded with service information and clients were pre-loaded with directory agent information. This step is not representative of the real situation. Mobile ad hoc networks are self configurable networks with no pre-programmed entries.

In addition, it will not be possible to represent the inherent characteristics of the elements of the network such as the periodic service advertisements from different servers, agent advertisements from directory agents, and clients receiving and updating their entries for the directory agent address. These assumptions can make the model invalid.

Moreover, it is assumed that all existing servers provide the same service. This is critical because the ability to differentiate among services contributes to service response time. Hence response time comparisons are less credible. Finally, there is no defined organization or grouping of nodes within the architecture. Consequently, there is a potential for network wide flooding during service discovery. This can seriously limit the scalability of the architecture.

3.4.3 Service Discovery in Mobile ad hoc networks

The study in [32] uses a proactive routing layer protocol OLSR (Optimized Link State Routing) for cross layered service discovery architecture. The OLSR protocol is extended with Mercury Service Discovery Message (MSD). The architecture uses a Bloom filter array as a structure to summarize several request messages or advertisement messages. Mercury handles requests and advertisements from two entities: local applications, and foreign nodes. Service advertisements are included in Bloom filters [23] and are flooded into the network using multipoint relay (MPR) flooding technique [43]. Similarly, if a service request from an application can not be resolved from the local service cache, a bloom filter of requested services is created and flooded into the network using the same MPR technique.

A major concern with this architecture is the presence of a frequent flooding of advertisements and request messages through the network. In fact, proactive routing protocols are not suitable for large ad hoc networks. The periodic control messages that are exchanged drain the power supply of mobile hosts. In addition, the bandwidth available for data traffic is limited. From these observations, it can be seen that the architecture can not scale to large networks.

3.4.4 AVERT: Adaptive SerVicE and Route Discovery ProTocol for MANETs

The research in [14] proposes a cross-layered architecture based on Independent Zone Routing (IZR) protocol. The protocol is a hybrid protocol that is derived from zonal routing protocol (ZRP). In the original protocol, nodes zonal radius, which is the number of hops the proactive route advertisements can propagate, is fixed for all nodes. Outside the zonal radius, a reactive protocol is used to initiate a bordercasting process to reach the node's boarder nodes (nodes at the border of the requesting node).

In real situations different areas within the MANET environment have different degree of call rates and mobility situation. Hence, a new adaptive protocol IZR was proposed. IZR allows each node to possess its own zonal radius which is dynamically adaptable based on the monitored traffic activity. The research compared the energy performance of AVERT with respect to IZR and SPIZ [44]. The Random Waypoint mobility model with 3.5m/s constant speed and zero pause time was used. The radio range was set at 380m. There were 20 nodes in the simulation study with three servers hosting different services.

However, factors such as periodic advertisement of services by servers within the zonal range, response time of service acquisition from servers outside the zonal radius or within the radius, are not discussed. The focus of the study was limited to energy consumption.

Chapter Four

PACL-SDP: Proximity Aware Cross-Layered Service Discovery Protocol for mobile ad hoc Networks

4.1 System Modelling and Assumptions

The system under consideration consists of **n** independent mobile nodes that can communicate by passing messages over a wireless link. This system can be modeled using graph theory as a set **G = (V, E)**. Where **G** is a graph, **V** is a set of vertices in the graph, and **E** is the set of edges of the graph [3]. An undirected graph is a graph where the vertices are connected by undirected arcs. MANETs and Wireless sensor networks, WSNs, can be modelled as undirected graphs [8]. The vertices of the graph represent the mobile nodes within the network, while the edges represent the communication links among the nodes.

The following assumptions are considered to describe the mobile ad hoc network under study in the context of distributed directories based service discovery architecture.

- Each node has a unique identifier (IP address).
- Communication links are bi-directional with FIFO queue discipline.
- The network topology is dynamic.
- Nodes can pass messages amongst themselves to create clusters within the network.
- A cluster has a unique leader at a time that serves as a directory node to coordinate service discovery processes within the cluster.
- Servers can periodically update services they offer by registering with a directory node.
- A directory node announces itself periodically within the cluster.
- Clients requiring services can send a service binding request message to a directory node if necessary.

4.2 Design principles of the new SDP architecture

PACL is a new cross layered service discovery protocol defined and designed in this thesis work. The routing layer protocol used for the implementation is Ad hoc On-Demand Distance Vector, AODV. AODV has been observed to exhibit overall superior qualities when compared to other routing protocols [45, 46]. It is the protocol that is characterized by low memory overhead, processing overhead, and network overhead.

Cross layering is a mechanism by which two or more processes at different layers of the networking stack are integrated to work more closely for the purpose of enhancing functionality of applications. In doing so, the traditional layered approach may be violated to some extent. However, this is in return for great benefit in terms of network performance.

In this new design the available features of the routing layer protocol have been exploited to facilitate the service discovery process. These features are the route request messages that are always broadcasted in search of a route to a destination and the HELLO messages that are used for neighbouring connectivity in the absence of any broadcast control message. As such, the integration for this new design involves the upper layers (Application and transport layers) and the routing layer.

There is also an attempt to incorporate some form of context in the discovery procedure. A context is any piece of information that can be used to characterize a process or an entity. It can be in terms of time, energy, space, etc. A process is referred to as context-aware if it uses context to provide relevant information. To that end, the context of physical proximity has been part of this new cross-layered SDP architecture. Thus, nodes can request for services at remote nodes that are two or more hops away only if their neighbours (nodes that are one hop away) can not provide these services.

Consequently, the architecture strives to localize interactions among nodes that are physically close to one another thereby limiting unnecessary network wide traffic within the environment.

There are two processes that PACL-SDP is based on. This research work has attempted to combine a leader election process with service discovery architecture. This step is justified as mobile ad hoc networks are self configuring autonomous networks that do not require interference from the user to setup the network environment.

The first process is the clustering or leader election process. At bootstrapping phase, the mobile nodes undergo a process of leader election where a node with more processing power in terms of processor speed and/or memory size is selected as the leader of the group. The leader election procedure is coordinated by an initiator node that begins the whole process. The first node that is aware of the absence of a cluster head can begin the procedure. The winner node then serves as the directory node or service repository for nodes within the cluster. The leader of the cluster announces its leadership to the members of the group by sending messages containing its unicast address that are piggybacked onto routing layer RREQ or HELLO messages. Receiving nodes update their entries of their cluster head. This capability stems from the integration of upper layer processes with lower layer process, yielding a cross-layered characteristic.

The second process involves the service discovery procedure. It is an implementation designed for MANETs based on IETF standard for service discovery protocols, SLP [29]. Nodes that possess services to offer periodically register their services by unicasting service registration messages to the cluster head that is serving as a registry node. The message contains the type of service offered, the address of the provider, service lease period, and optional service attributes. These service registration procedures are issued at the upper layer protocol by passing the registration message down to the routing layer.

On the other hand, every node within the network that possesses some service uses the already existing routing layer broadcast traffic to advertise its services to its immediate neighbours.

This announcement is piggybacked onto the RREQ or HELLO routing messages without expending any extra processing power for service announcement. This also complements the cross-layered feature of the protocol. On top of this it helps nodes to be aware of services that are in close physical proximity, hence asserting the physical proximity awareness feature of the protocol. Thus, a node requiring a service first consults its neighbours' service cache before sending a service inquiry message to the directory node of the cluster.

4.3 The leader election process

In this work a leader is not randomly elected. Nor could we follow a certain mathematical algorithm based on the spatial distribution of the nodes to serve as the primary reason to select one node or another as the leader. Instead, the chosen preferred criterion that is used in this work to qualify a node to be a winner as a leader is the amount of processing power (memory size and processor speed) it possesses at the time of the election process. It is worth mentioning, however, that there exist several other factors that can be incorporated for comparison to elect the appropriate node. Among these additional factors one can mention available energy, speed of node, operating system type and version, etc. A numerical weight quantity can be associated with each factor that signifies its importance for a situation. The computed score value can then be used to elect the most valued node. For simplicity, in this work only processing capacity is considered. Therefore, here the node with the largest processing power wins the election. This is required to allow the resource rich node to serve as the cluster head. The cluster head will have the responsibility of hosting service related information within its domain or territory and responding to service queries from clients thereby playing the role of a directory node.

For the proposed algorithm it is assumed that the communication links are bidirectional with FIFO queue discipline. It is also assumed that nodes have unique identifiers.

The process of leader election algorithm can be explained as follows. The first node that has learnt the absence of any leader can initiate the whole process. Such a node (called the INITIATOR) broadcasts a leader selection packet (LS-packet) that is set to reach a diameter of n hops from the INITIATOR. Here a diameter of 2 hops is used. Hence, a cluster is formed by nodes that are up to two hops away from the INITIATOR node. It is assumed that any node can play the role of an INITIATOR. The contents of the leader selection packet, LS-Packet, are shown in Figure 4.1.

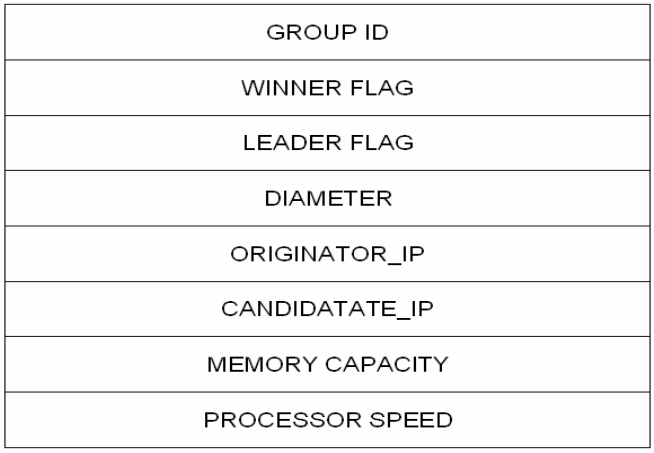

Fig. 4.1 The Leader selection (LS) packet

The fields contained in the packet can be explained as follows:-

- *Group ID*: (integer) represents the cluster identity which is a randomly generated number set by the INITIATOR node.
- *Winner Flag*: (integer) represents the flag that is set by the INITIATOR node when a winner node is known. A node receiving an LS packet with Winner Flag set learns that it has won the election process.

- *Diameter*: (integer) Controls how far the designated packet propagates in the network.
- *Origninator_IP*: (4 bytes) Specifies the unicast address of the INITIATOR node.
- leader flag: (integer) Used by a winner node to notify its lower layer that this node is a leader node.
- *Candidate_IP*: (4 bytes) Any node that modifies the processing capacity of the LS-packet enters its unicast address here. This helps the INITIATOR to know which node possesses the claimed processing capacity.

- *Memory Capacity*: (integer) Used by a node to record its memory size in MB.
- *Processor Speed*: (integer) Used by a node to record its processing speed in MHz.

The LS packet contains fields that specify the processing capacity of the node. In this study memory size and processor speed are used. The INITIATOR broadcasts the packet to its one hop neighbours after recording its processing capacity. The packet also contains a diameter parameter that sets the number of hops the message is to be re-broadcasted. Any node that receives the broadcast compares its own capacity with the contents of the LS-packet. The action to be taken by the node depends on two factors. The first is the result of the comparison of the computational capacities. The second is the distance (number of hops) the node is away from the INITIATOR node. Consider Figure 4.2 to see how a node reacts based on the above two factors.

In any case, if the node has a better capacity, it modifies the contents of the LS-packet that specify the processing capacity. Any node that has included its processing capacity in the LS packet, records its IP address in the CANDIDATE filed of the packet.

This helps the INITIATOR node to know which node possesses the claimed processing capacity. Next, depending on its distance from the INITIATOR node (which it learns by reading the diameter field) it will either broadcast the modified packet or unicast the new packet to the INITIATOR node.

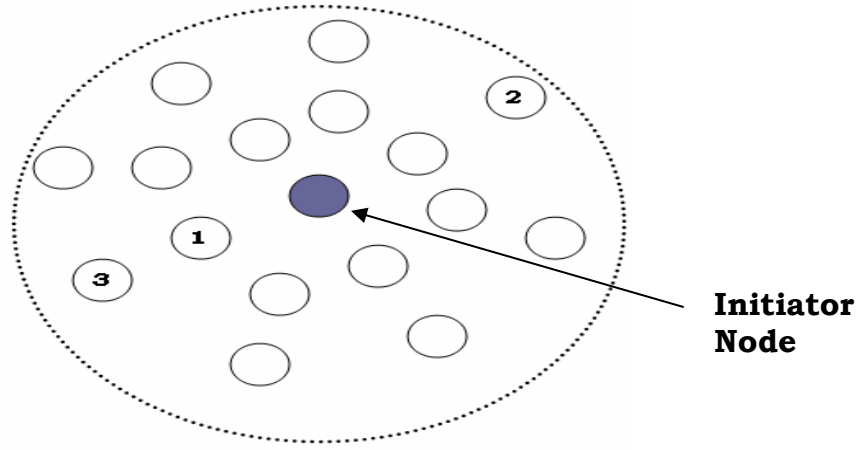

Fig. 4.2 The leader election process

If the node finds itself one hop away from the INITIATOR node, for example at position 1 in the figure, it will broadcast the modified LS-packet so that the INITIATOR node and those other nodes that are two hops away from the INITIATOR can receive the message. It also decrements the diameter field by one. Otherwise, if the node finds itself two hops away from the INITIAOTR, position 2 in the figure, it will unicast the modified packet to the INITIATOR.

Now what happens if contents of the generated LS-packet had superior capacity than the processing capacity of the receiving node? Again, the action to be taken by the node depends on its position from the INITIATOR node. If it is one hop away from the INITIATOR node, position 1 in the figure, it will re-broadcast the packet without modifying the processing capacity fields. It will only decrement the diameter field by one. The necessity of this step is clear by now. It allows the other members of the cluster (which are two hops away from the INITIATOR), such as node 3, to receive the packet and take their own decision by comparing the contents of the packet with their own processing capacity.

On the other hand, if the node is a leaf node of the cluster (i.e. it is at the perimeter of the cluster), consider position 2 in the figure, the receiving node does not take any action. It simply drops the packet. This is required to avoid any unnecessary unicasts of the leader selection packets by those nodes with inferior capacity. This step conserves both bandwidth and energy. Finally, the INITIATOR node uses a time out limit after which it will announce the winner node based on the received broadcasted and unicasted messages until the timeout period. Note that if the INITIATOR node does not receive any response until the timeout period, then this node is an isolated node, i.e. not part of the ad hoc network.

Here it is worth mentioning that there is the possibility to nominate additional secondary and tertiary nodes as well for standby purposes. However, this feature is not included in this work.

The winner node begins sending a modified routing layer HELLO or RREQ messages that contain information about its leadership and its unicast address. These pieces of information are periodically included in the HELLO messages for neighbouring management or whenever the leader participates in route request broadcast messages. Hence, the leader announces itself using already existing routing layer messages thereby saving extra energy or bandwidth that would have been used. This cross-layering feature is unlike most other leader election algorithms that would use upper layer messages to maintain their leadership. In any case either of the messages that are generated will piggyback the leader information. Nodes that receive these messages will update their entry for the cluster head and propagate the message in their next broadcast message which can be again either a modified HELLO or RREQ message.

The flow chart diagrams below show the basic processes of the leader election process. Figure 4.3 depicts the initial process as generated by the INITIATOR node.

Part (a) of the figure represents the main procedure executed by an INITIATOR node while parts (b) and (c) show the service routines invoked, respectively, when an LS packet is received and when the timeout period expires.

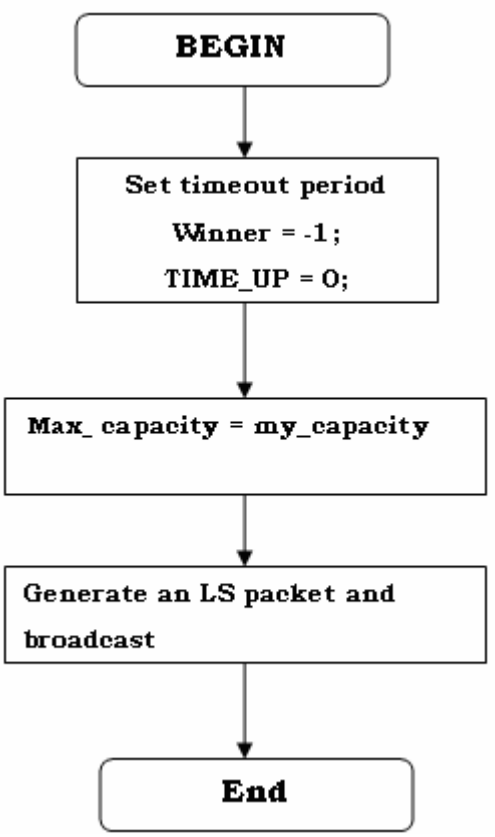

Fig. 4.3 (a) Main procedure in the INITIATOR node to begin the LS process

As can be inferred from Figure 4.3 (a), the INITIATOR node first sets the variable indicating the potential winner node, winner, to a value of -1. This is useful because at the time when the timeout period expires, if the value of the variable is still -1, then the INITIATOR node learns that it is an isolated node.

From the same figure, the variable TIME_UP is used to avoid the processing of any late incoming LS packets. The INITATOR node can not be waiting indefinitely for LS packets. Once the timeout period is fired, the INITIATOR node sets the TIME_UP flag up as can be seen from Figure 4.3 (b). Next, it will announce the winner based on the received packets thus far.

If there are any packets that arrive later, those packets are dropped. Note also that the INITIATOR node assumes its own processing capacity as the value of the variable, Max_capacity. This variable serves as a place holder of the capacity for the potential winner node.

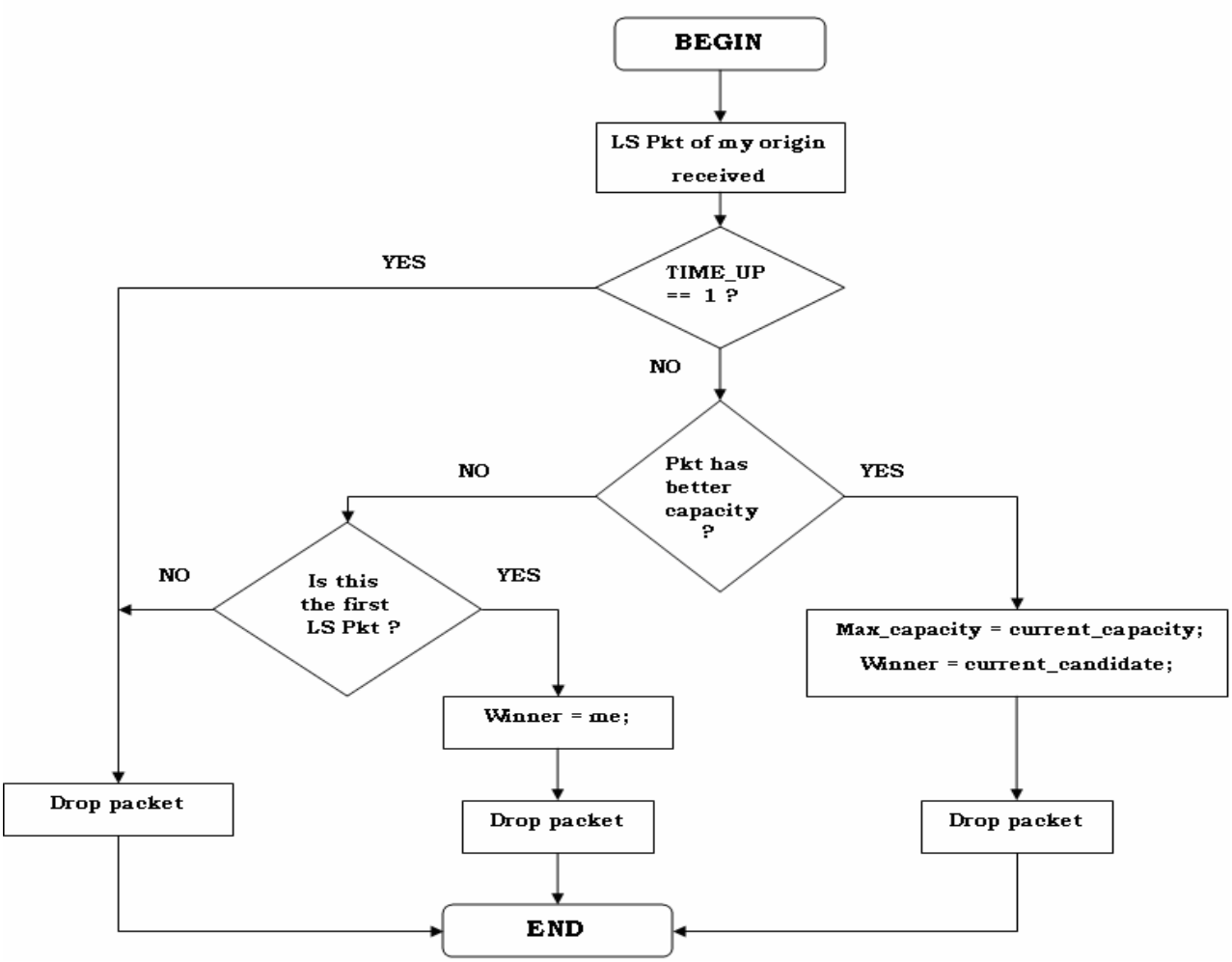

Fig. 4.3 (b) Service routine of INITIATOR node when an LS packet is received

A close examination of Figure 4.3 (b) shows that an INITIATOR node that possesses the highest processing capacity can nominate itself as the potential leader of the cluster only if the node receives at least one response from its neighbours. At that moment, the INITIATOR node sets the winner variable to its address, as can be seen from the figure, “winner = me”. This is an indication that the node is within the radio range of at least one other node.

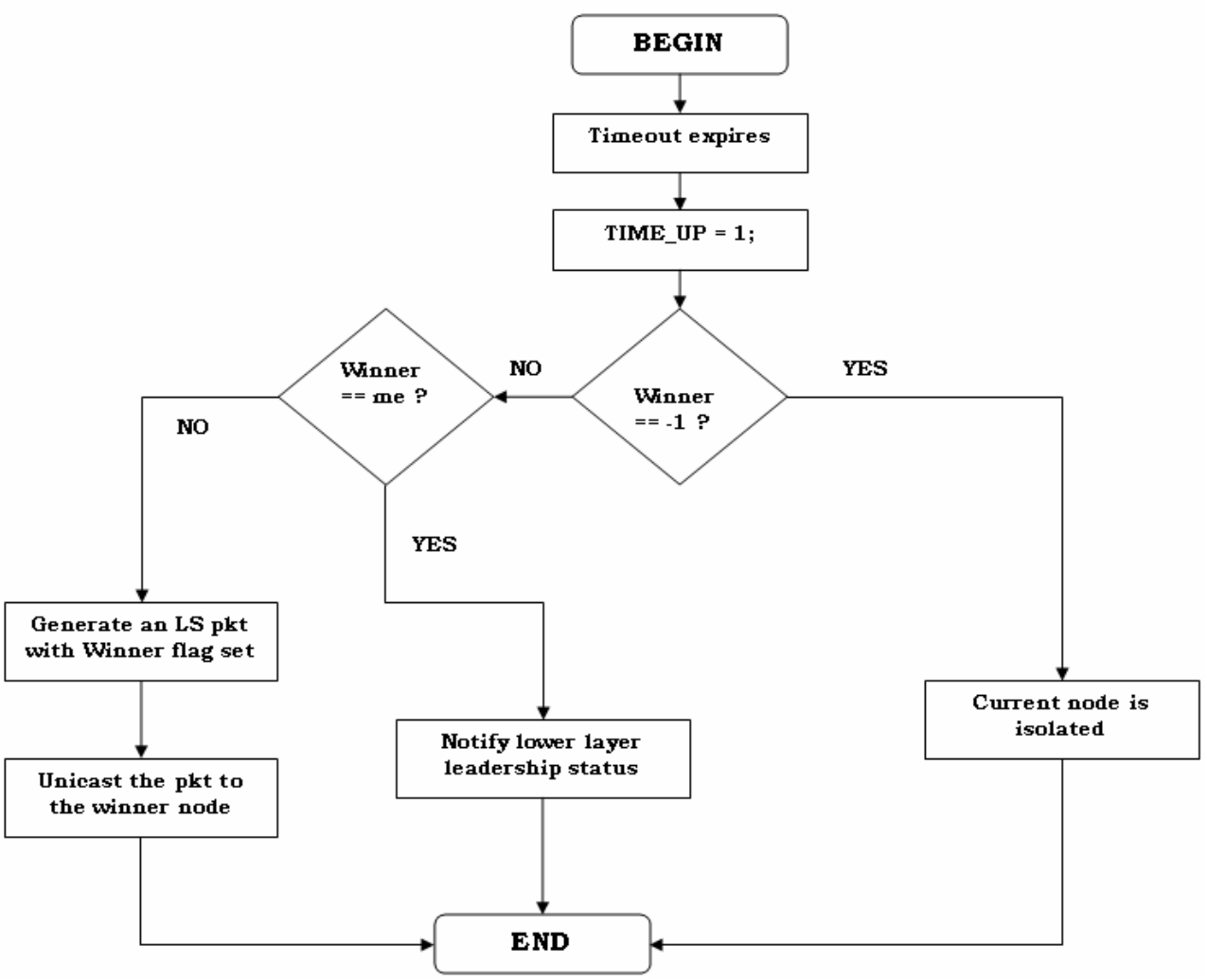

Fig. 4.3 (c) Service routine of INITIATOR node when the timeout period expires

All the processes illustrated in Figure 4.3 are initiated and executed at the upper layers. The involvement of the routing layer is postponed until the INITIATOR node notifies the winner node of the group. When a node receives a unicast LS packet coming from the INITIATOR node with the winner flag set, it alerts its routing layer to use modified control messages (HELLO messages or RREQ messages) containing its leadership advertisement.

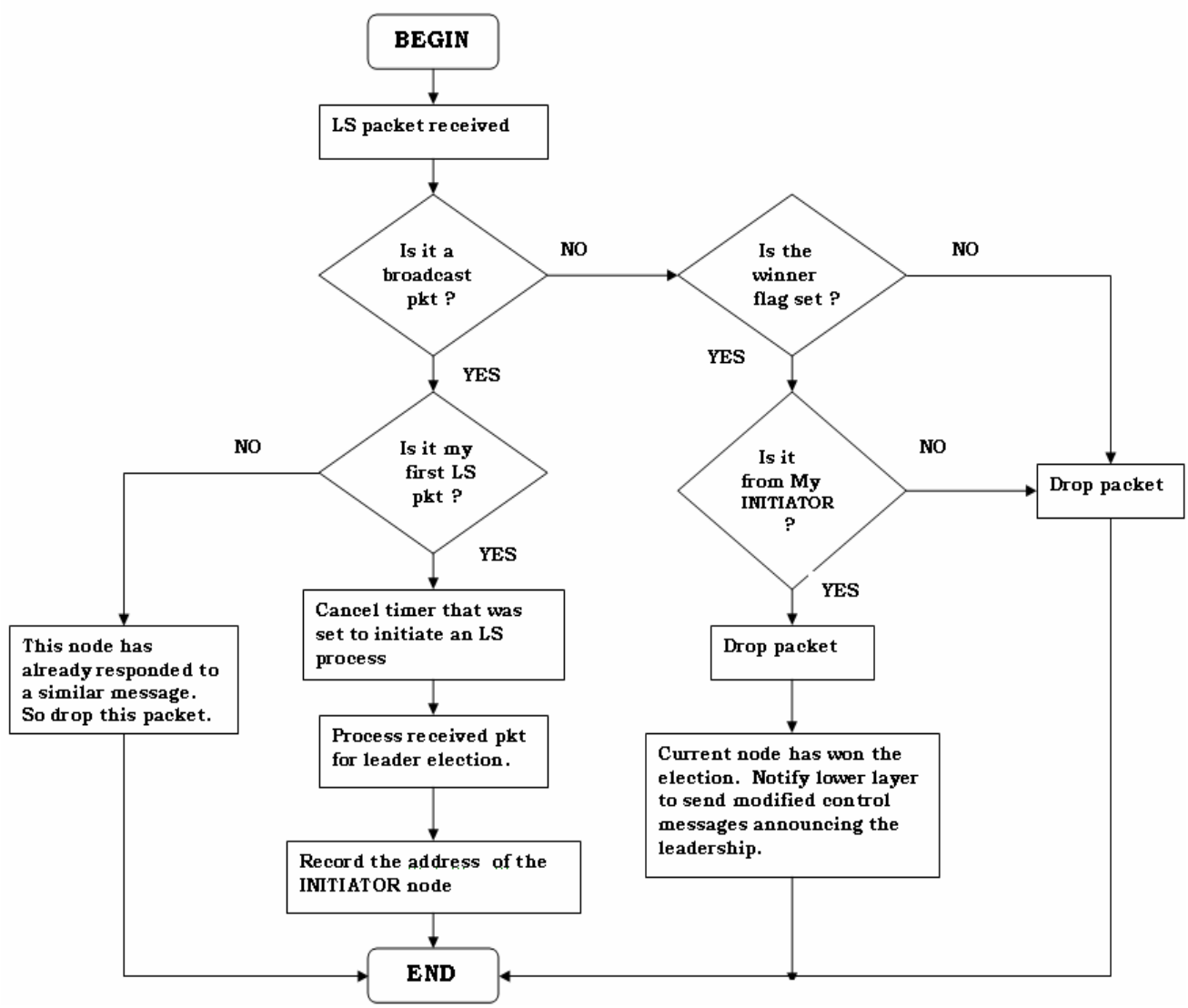

Fig. 4.4 Service routine when a member node receives an LS packet

Figure 4.4 illustrates the service routine that shows the actions to be taken by a member node when receiving the leader selection packet. Observe that when a node receives its first LS packet, it has to cancel its timer that is used to initiate a leader election process. Should this step fail, the node would generate another leader election process when this timer expires.

4.4 The service discovery process

In the architecture of the new cross-layer design there exist three types of service caches. These are the local service cache (LC), the neighbours' service cache (NC), and the directory cache (DC).

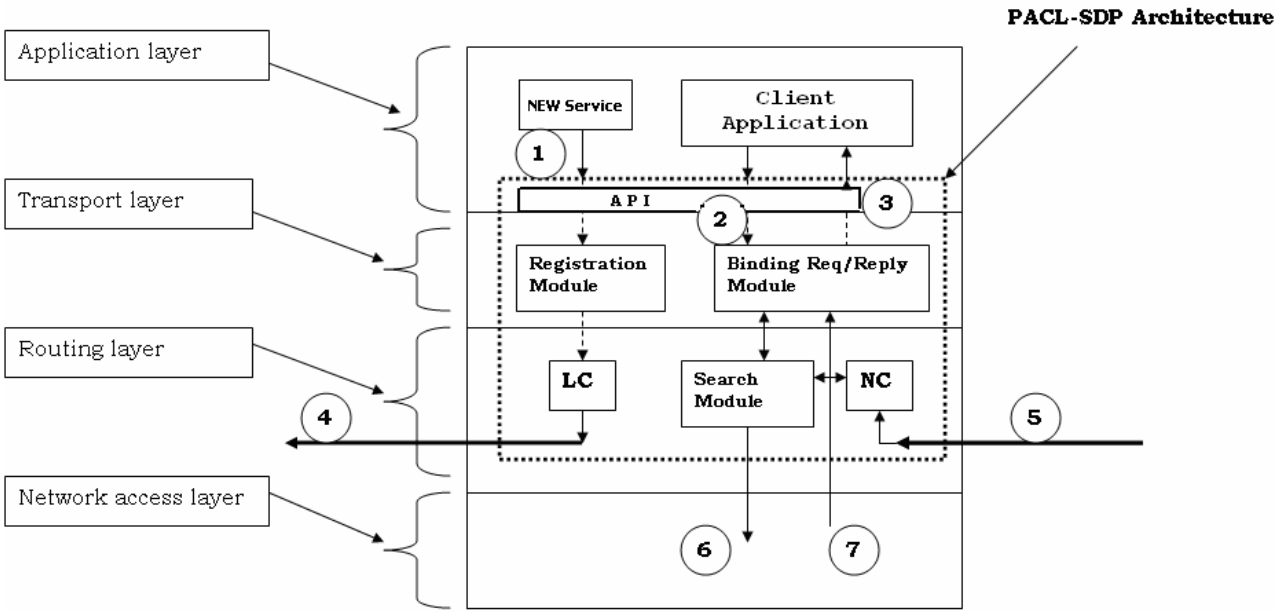

Fig.4.5 PACL-SDP architecture of a mobile node

The local services cache exists on all nodes. It is used to register services available on the local host. As can be seen at position 1 in Figure 4.5 whenever a new service is installed on a host, the host registers the new service binding, the association between the service type and the IP address of the host offering the service, in its local cache. This cache is consulted every time the node advertises its services to its neighbours using RREQ or HELLO messages as shown at position 4 in the figure.

The second type of cache is called neighbours' cache or members' cache. It also exists on all nodes. It is used to store service binding advertisements coming from neighbouring nodes as illustrated by position 5 in the figure.

The third type of cache is the cache found at the directory cache. This is infact basically the neighbours' cache for the directory node. It is used to store service bindings coming from the remote nodes (members of the cluster) as well as neighbours of the directory node. The three types of caches can have the structure shown in Figure 4.6.

Fig. 4.6 PACL-SDP service cache

Service name Gives the type of service (eg. temperature sensor, printer, chat server, etc)

Server address: Contains the unicast address of the node hosing the service.

Service lease period: Records the duration period for which the service will be available.

Service attributes: This is optional field that contains additional information for the service. For instance a colour printer, laser printer, etc.

As illustrated in Figure 4.5 when an application (or a user) requests for a certain service not found on the host, consider position 2 in the figure, the upper layer passes a binding request message to its lower routing layer. The lower layer consults its neighbours' cache for service binding. If a binding is found, then the routing layer issues a call-back to the transport layer delivering the binding reply as indicated by position 3 in the figure. From this step onwards, the upper layer operates following the normal procedures an application uses to invoke a service.

However, if no binding is found at the neighbours' cache the routing layer generates a service binding request message towards the directory node of the cluster. This is represented by position 6 in the figure. Position 7 shows an incoming service binding reply coming from the directory node.Depending on the packet type filed, a service discovery packet for PACL-SDP, can serve different purposes. Figure 4.7 shows the general packet format for a PACL-SDP service discovery packet. One such purpose is when the packet is used to carry a service binding request or a service binding reply message. The other types and their purposes are given below.

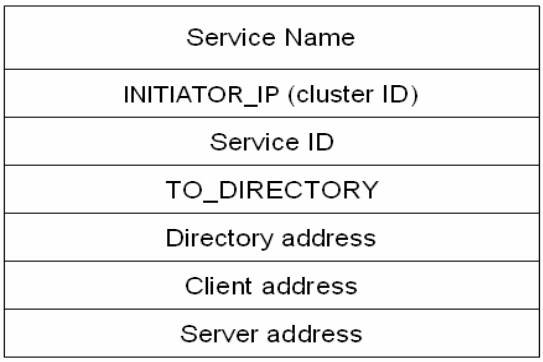

Fig. 4.7 PACL-SDP packet

- Service Name (integer): Contains the type of service
  (Sensor, gateway, file server, etc)
- INITIATOR_IP (4 bytes): Defines the cluster ID that uniquely identifies the cluster from which the packet is originated
- Service ID (integer): Identifies the service discovery packet function.
  It can be:-
  service binding request/reply to/from directory (type 4),
  service binding registration at directory node (type 5)
  service binding registration at local cache (type 1)
  service binding request by upper layer (type 2)
  service ack response by destination server node (type 3)
  service binding reply from neighbouring cache (type 6)

- TO_DIRECTORY (integer): When set tells a client node to send message to Directory node.
- Directory address (4 bytes): Identifies the required Directory node.
- Client address (4 bytes): Specifies the unicast address of the host requesting the service.
- Server address (4 bytes): Specifies the resolved IP address of the server hosting the service.

The service discovery packet with type 3 is used to learn the existence of a valid service response from a resolved server. It can also be used to get information on the amount of time the service requester has waited before it gets the first reply from the server. This is referred to as the response time of the process.

The flow chart diagram presented in Figure 4.8 show the steps an ordinary node follows when using the service discovery protocol. The flow chart shows that an ordinary node may invoke PACL-SDP for either of two reasons. The first is when an application/user requires a service that is available on another node. The second possibility is when a new service is installed on the node, the node has to register this service in its local cache.

Figure 4.9 shows the basic processes of the service discovery process for a directory node. As can be inferred from the figure, a directory node may invoke PACL-SDP for two additional reasons other than those defined above for the ordinary node. The first is when a server registers its services at the directory node. The second possibility is when a client node sends the directory node a service binding request message.

It is not the scope of PACL-SDP in this study to include how service invocation is done. Nor does it specify any specific service description techniques to follow. There are features of a service discovery protocol that are not dependent on the architecture of choice. Among these features are; service description techniques, mobility support, and service selection mechanisms [1]. Consequently, a given architecture can use any service description technique. The most widely used is extensible markup language (eXML) and its derivatives. It is also possible to use simple text attribute-value schemes. The latter is used in this work.

In order to carryout its functions as stated, PACL-SDP requires that the necessary modifications be made at the routing layer. The underlying network layer protocol that supports PACL-SDP is the Ad hoc On-Demand Distance Vector, AODV. Hence, modifications on this protocol have been carried out both at packet level and process level. The added fields in the control packet structure and the accompanying process-level implementations are subjects of the final sections of this chapter.

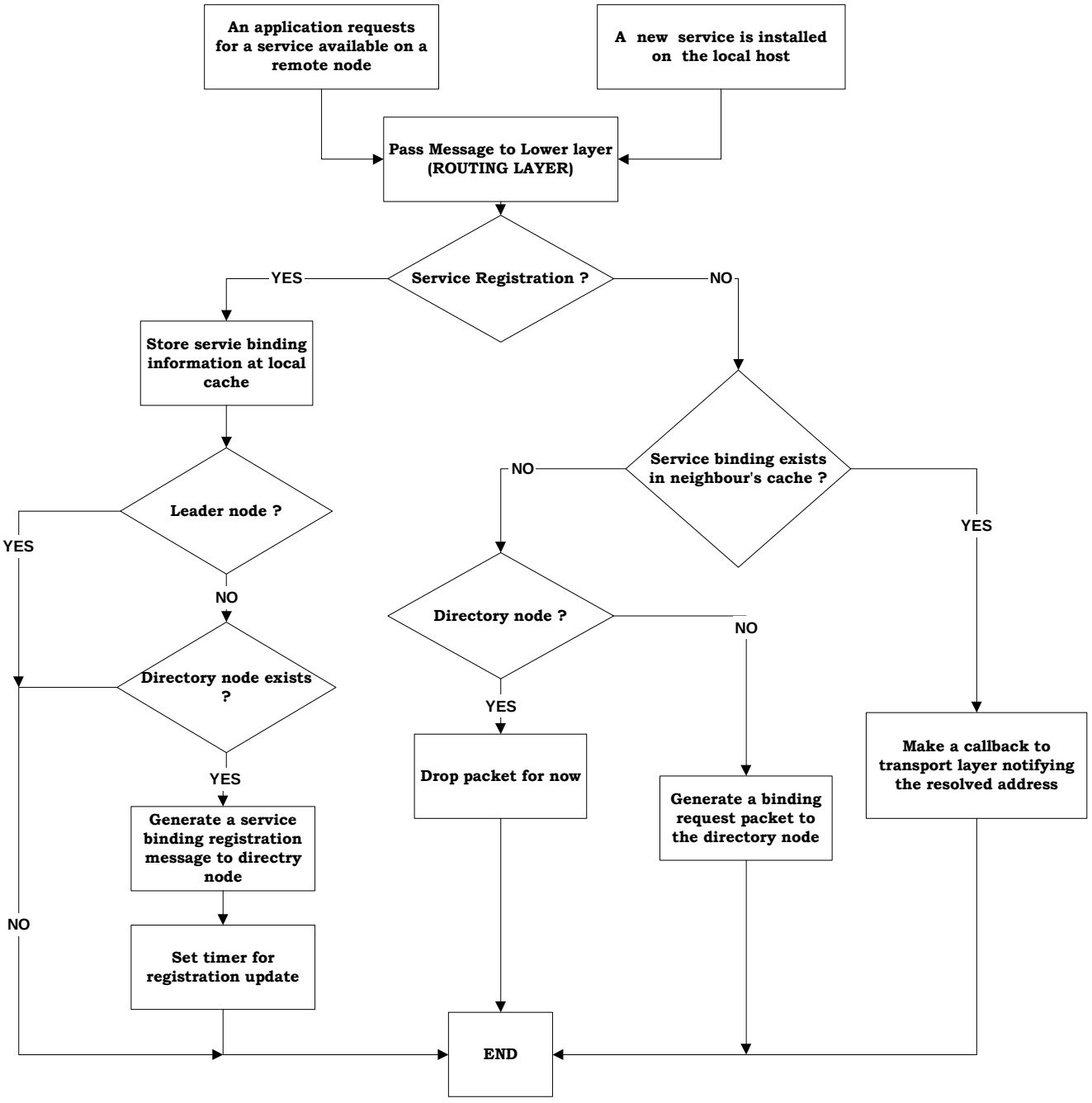

Fig. 4.8 PACL-SDP process by a typical node

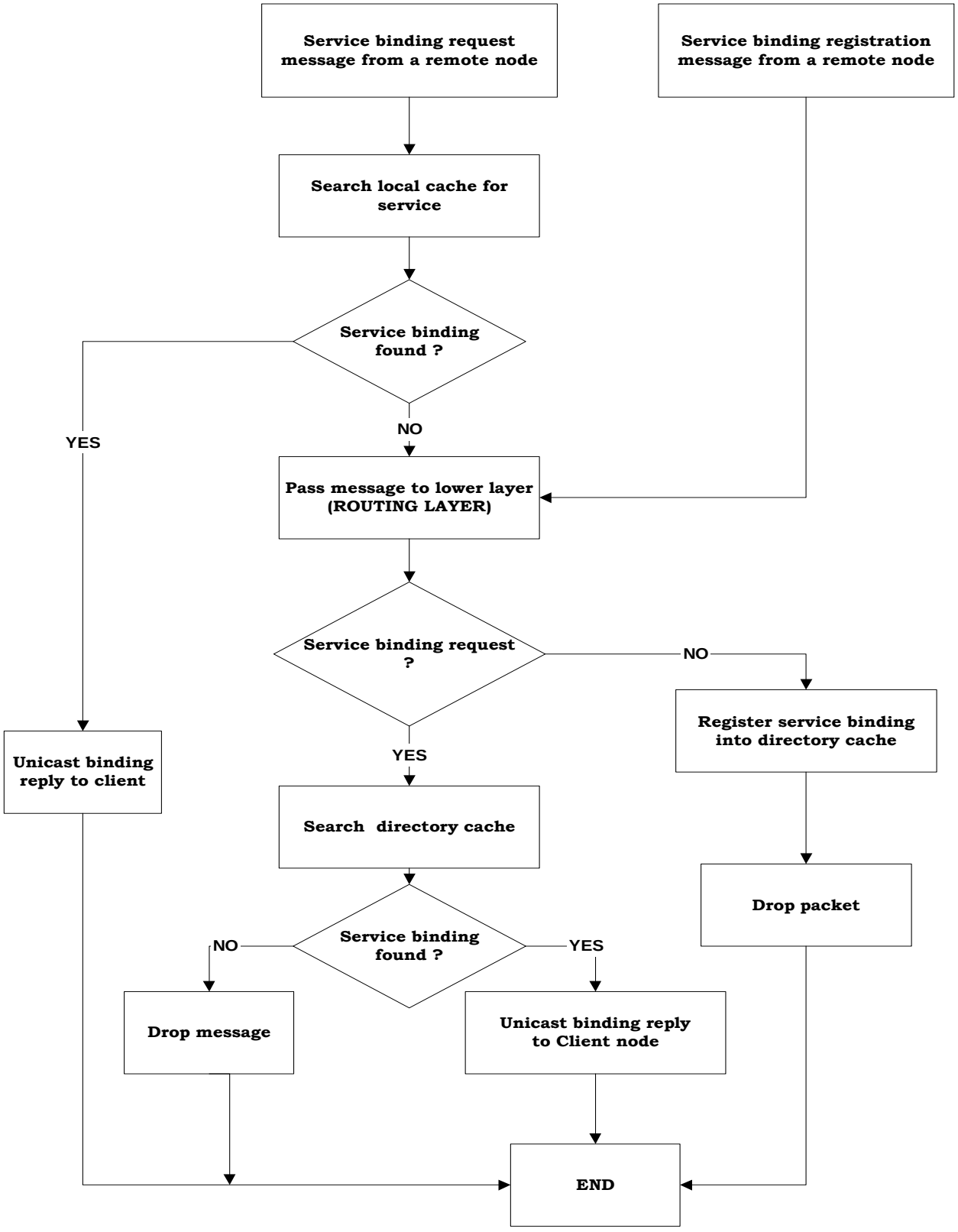

Fig. 4.9 PACL-SDP process by a Directory node

4.5 Packet and process level modifications at the routing layer

The AODV routing protocol has been modified to facilitate the service discovery process. The control packets chosen for the step are the route request (RREQ) message format and the HELLO message format. The new fields added to these packets enable the routing layer process to piggyback service related information whenever these messages are issued. The required pieces of information needed to be carried over are directory address announcement and service bindings from local hosts. It was required to implement the modifications at both types of control packets because a mobile node may not generate a HELLO message if it has recently generated a RREQ message. Similarly, if no RREQ message has been issued within a HELLO INTERVAL (the period set for HELLO messages), then the mobile host issues a HELLO message [33]. Which control message is generated depends on the activities of the nodes within the network. For instance, for a network with few nodes HELLO messages are more prevalent. And for a network with relatively large number of nodes RREQ messages are more prevalent. This justifies the need to carryout the packet-level modification at both types of control messages.

The figures below illustrate the changes made at the RREQ and HELLO message packets. Figure 4-10 shows the extensions made for RREQ packet. The shaded portion represents the newly added fields to incorporate the cross-layering features. The added fields can be explained as follows:-

- *dim*: Specifies the number of hops the service binding information is allowed to propagate into the network
- *m_flag*: Specifies whether the packet is modified or not. If set it signifies the packet is a modified control message carrying header address.
- *service_code*: Records the type of service that is carried by the message

- *not_empty*: A field that specifies for the recipient of the message that there is service binding.
- *Cluster head IP address*: The unicast IP address of the directory node.
- *INITIATOR IP address*: The unicast IP address of the INITIATOR node. This is used as a cluster identifier.
- *Server host IP address*: The unicast address of the node advertising the the service.
- *Service Lease period*: The duration period for which the service will be valid before service renewal.

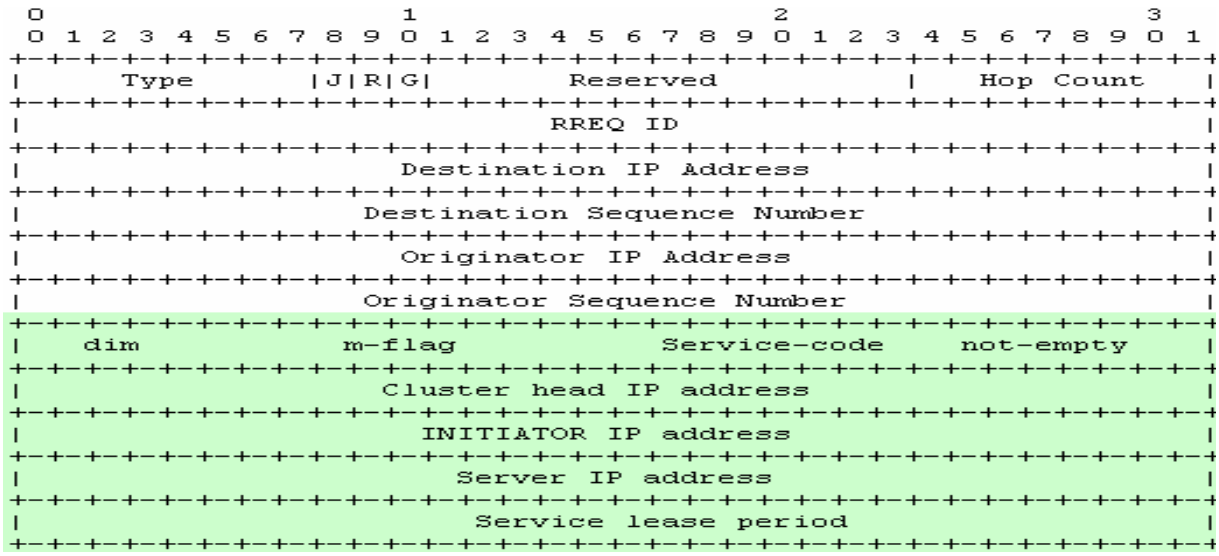

Fig. 4.10 Modified RREQ message format

The modification carried out for the HELLO message is identical. The field parameters described above equally apply this message format too.

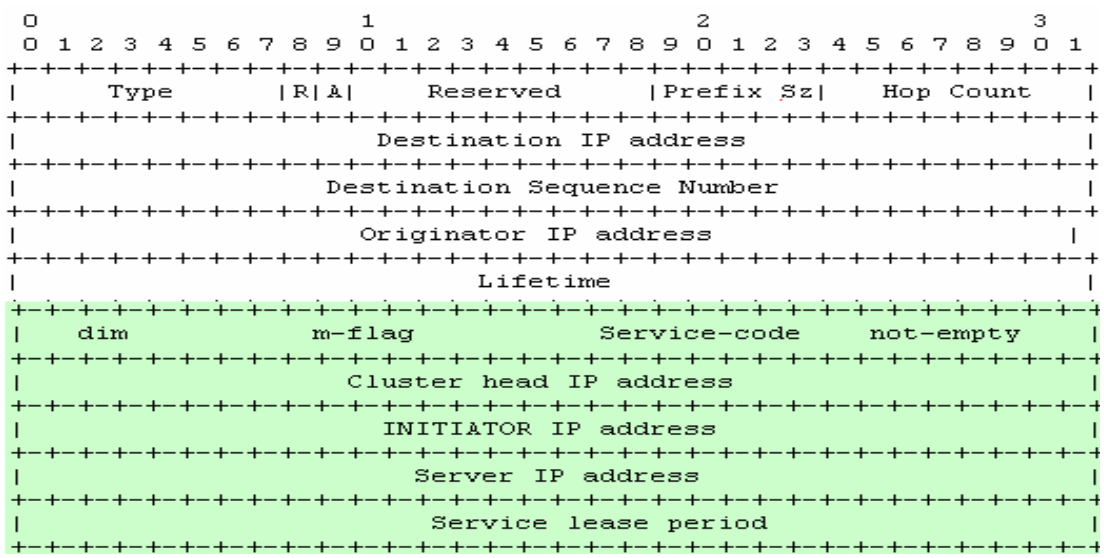

Fig. 4.11 Modified HELLO message format

The implemented modification in the process flow rule of the AODV algorithm is illustrated in the flow chart diagram presented at Fig. 4-12. The flow chart shows the steps to be undertaken by the receive and send modules of the AODV protocol to handle directory (leader) as well as service related information.

From the discussions presented thus far, it can be observed that the new service discovery protocol has the following features:

- It has a distributed architecture
- It is cross-layered
- It is context aware (aware of physical proximity and service lifetime)
- It is highly scalable (flooding of service query messages is avoided)

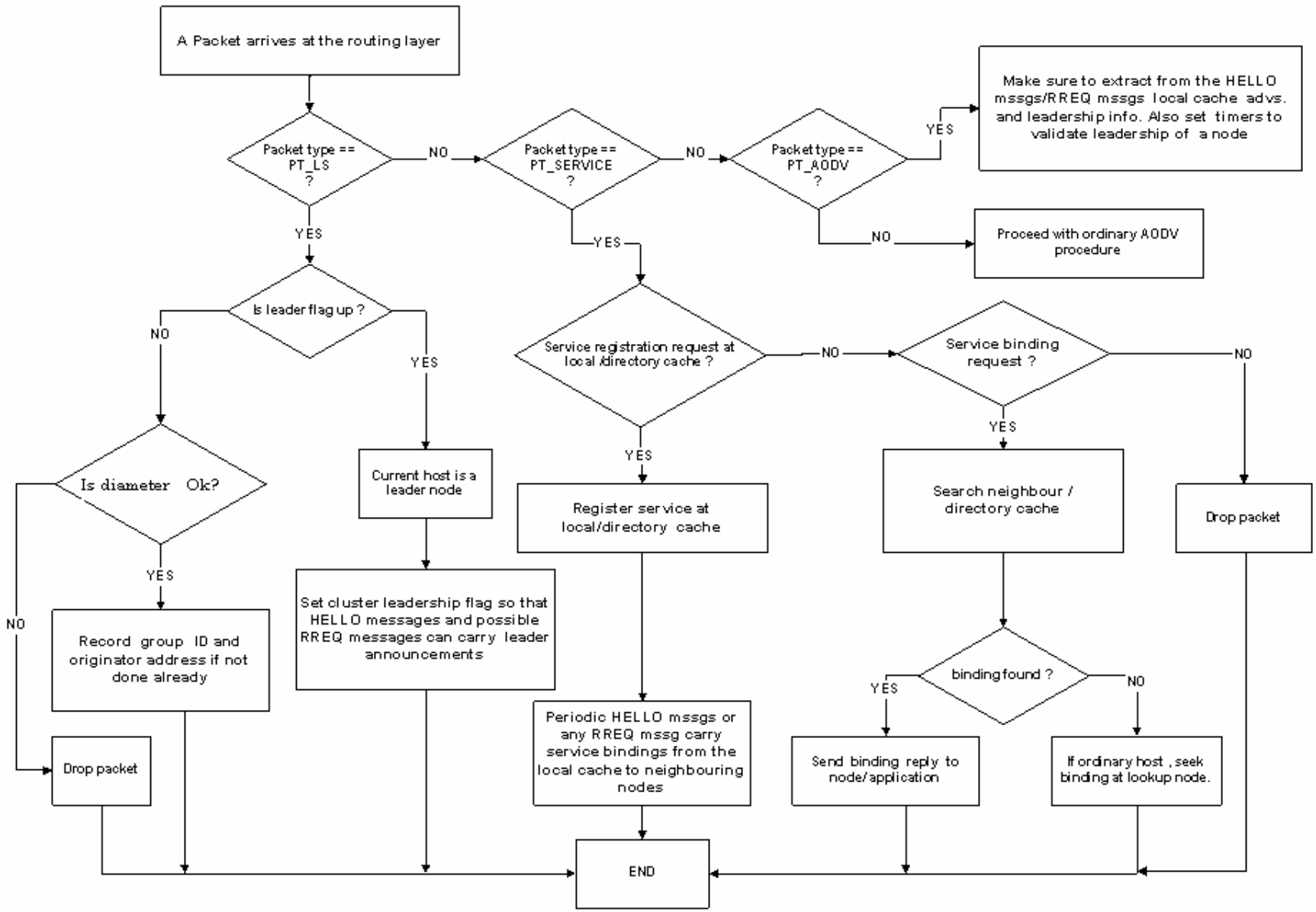

Fig. 4.12 Flow chart of process implementation included to modify the AODV protocol

Chapter Five

Simulation Results and Analysis

This chapter begins by briefly assessing the mobility models used for the study. It is followed by detailed description of the implementation process for the PACL-SDP. The implementation was carried out with simulator NS 2 ver. 2.34 [15] on an open SUSE platform as NS 2 is the most widely used simulator that is acceptable by engineering and scientific researchers. The conclusion of the chapter presents the comparative analysis of the new cross-layered service discovery protocol with the directory based SLP_MANET, which is an application layer based distributed service discovery protocol for mobile ad hoc networks implemented using the SLP standard from IETF.

5.1 Mobility models employed in the analysis

The elements of a mobile ad hoc network are usually mobile in nature. Consequently, in studying the behaviour of such a network usually a model that simulates the mobility of the elements is used. This model is simply referred to as the mobility model. The value of evaluating a mobile ad hoc network protocol is highly dependent on how realistic is the movement pattern generated by the mobility model.

There are two different ways to get movement patterns for mobile hosts. One source of patterns is a collection or data bank of real traces where actual movement patterns of mobile nodes in the hands of their owners are recorded and used for the study. These are what refered to as realistic traces. The other type is the use of synthetic traces generated by mathematical models [9, 10]. Thus, using realistic mobility models for simulation environments are preferred to synthetic models. However, real traces have not been widely used for evaluation because the available data are limited.

Moreover, those that are available are related to very specific situations and as such their validities can not be generalized. Hence, synthetic mobility models are widely used in evaluation of network protocols for mobile ad hoc networks.
Synthetic mobility models can be broadly classed into two types [9]. Entity mobility models, where the mobile hosts move independently of one another, and group mobility models where the mobile nodes movements are dependent on each other.

The next sections highlight the synthetic mobility models used in this thesis work. Two entity mobility models and two group mobility models are used in this research work.

a) Entity mobility models
    i) Random Walk mobility model
    ii) Random Waypoint mobility model
b) Group mobility models
    i) Reference Point Group Mobility (RPGM) model
    ii) Community based mobility model

5.2.1 Random Walk mobility model

The random walk mobility model was first described mathematically in 1929 by Einstein [9]. The model was developed to mimic the unpredictable erratic movement of entities in nature. The random walk mobility model represents the equivalent of Brownian motion. In this mobility model, a mobile host moves by choosing its speed and direction randomly. Each movement occurs either in constant time interval or constant distance travelled, at the end of which a new set of speed and direction is chosen and the process continues until the end of the simulation time.

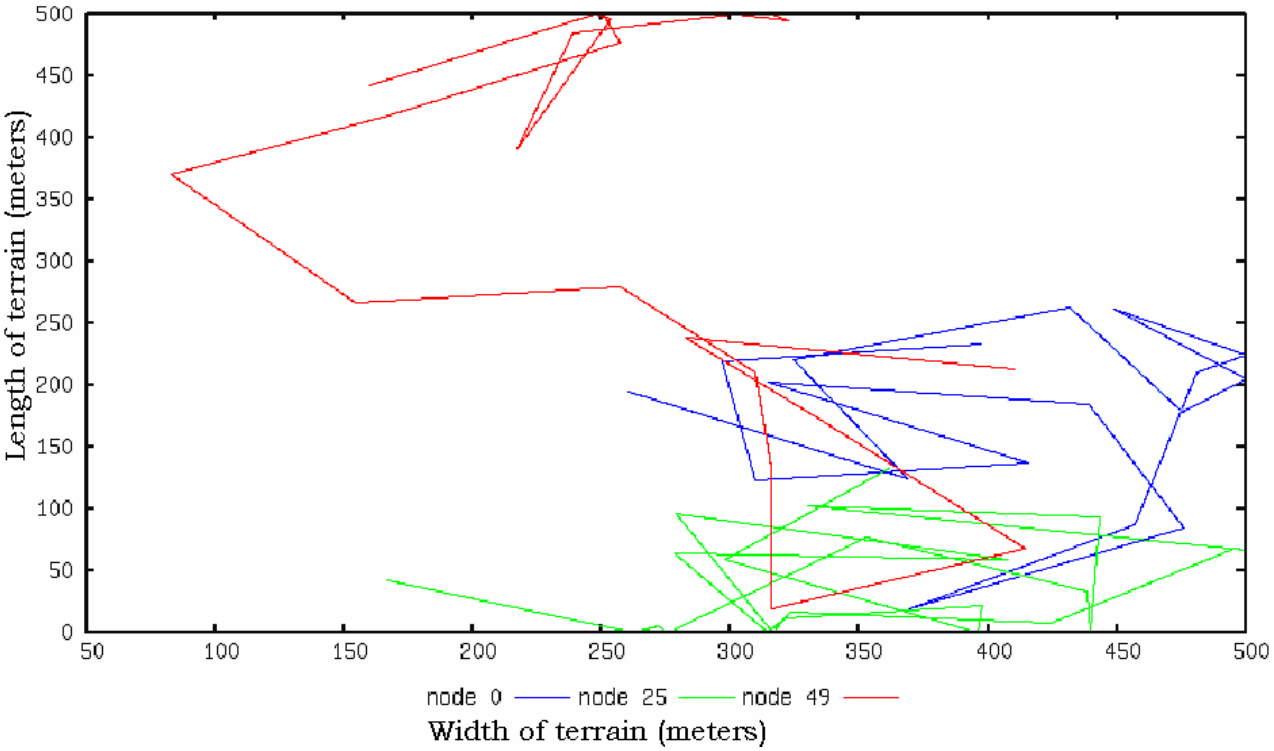

Fig.5.1 Travelling pattern of 3 mobile nodes with random walk mobility model

Figure 5.1 illustrates travelling pattern of three chosen nodes among 50 mobile hosts moving with average speed of 3.5 m/s and speed delta 1.0 m/s for 500 sec simulation time.

5.2.2 Random Waypoint mobility model

The random waypoint mobility model is a slight modification of the random walk mobility model. It includes pause times between changes in direction or speed [9]. In most of the performance evaluations nodes following the random waypoint mobility model are randomly distributed over the simulation area. This is not representative of real life systems.

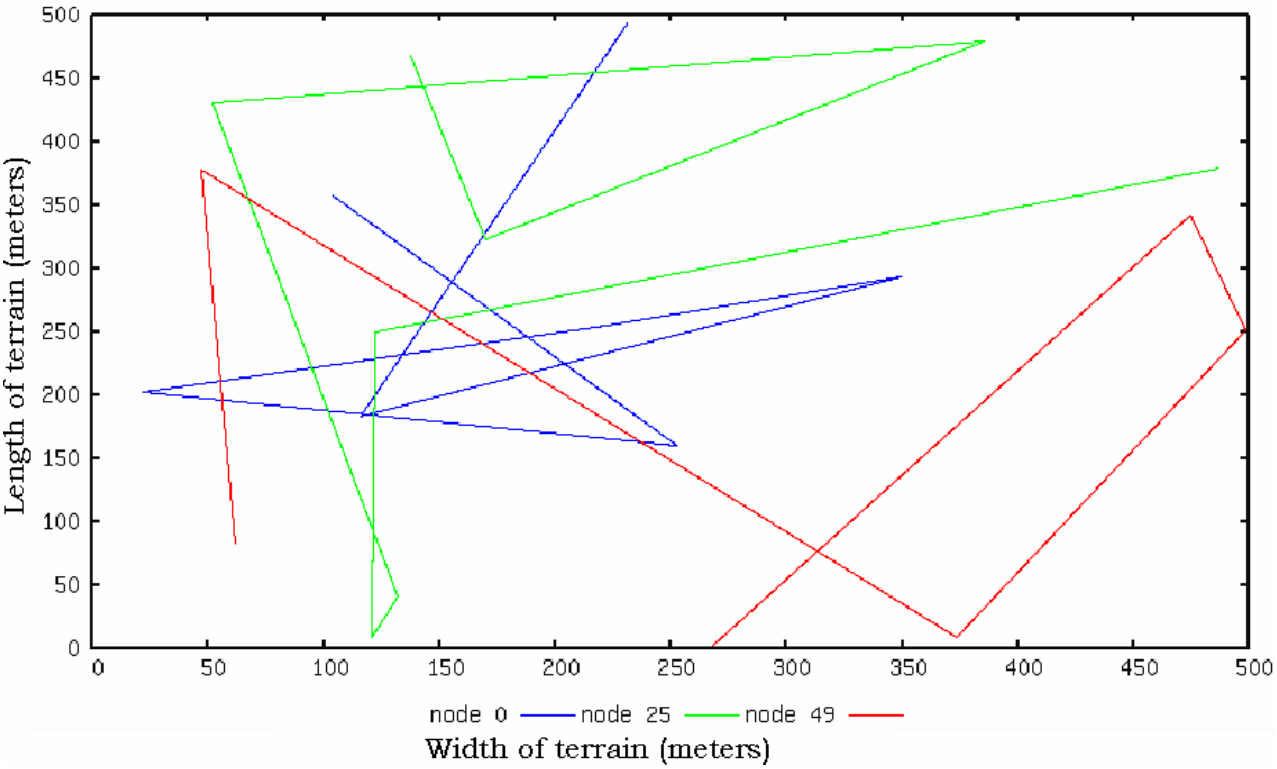

Fig.5.2 Travelling pattern of 3 mobile nodes with random waypoint mobility model

The illustrated screen capture in Figure 5.2 represents the travelling pattern of three mobile nodes out of 50 nodes with average speed of 3.5 m/s and speed delta of 1.0 m/s over a simulation period of 500.0 sec. Pause time used was 20.0 sec with pause delta of 5.0 sec.

There exist, however, serious critics on the validity of random mobility models to represent real life human networks. The authors in [10] mentioned that in general random mobility models generate patterns that are most unhuman like. They indicated that mobility models based on random mechanisms generate traces that show properties (such as the duration of the contacts between the mobile nodes and the inter-contacts time) very distant from those extracted from real scenarios.

In mobile ad hoc networks there are several situations where a group of mobile nodes move together for cooperative work. To mention few of these instances one can consider a group of soldiers moving together to destroy land mines, a conference of people dividing into groups to deal with different issues, in disaster relief systems where medical staff of a department work together, etc. In order to model such situations a group mobility model is needed to represent such cooperative nature. The next two sub sections introduce two group mobility models that can be used in the study of mobile ad hoc networks.

5.2.3 Reference Point Group Mobility (RPGM) model

RPGM represents the random motion of a group of mobile hosts as well as the random motion of individual nodes within the group [9]. Group movements are effected by the movement of a logical centre of the group. The motion of the group centre characterizes the motion of its member nodes including their speed and direction. Individual nodes move randomly around their pre-defined reference points, whose movement is dictated by the group movement. The movement of both the logical centre as well as the individual nodes is using the random waypoint mobility mode. However, individual nodes do not use pause times while the group is moving. Pause times are used when the group reaches a certain destination where all nodes use the same pause time.

The screen capture in Figure 5.3 demonstrates the movement pattern of three mobile hosts with RPGM model with average speed of 3.5 m/s with pause time of 20.0 sec and pause delta 5.0 sec. The speed delta used was 1.0 m/s. The simulation period was 500.0 sec.

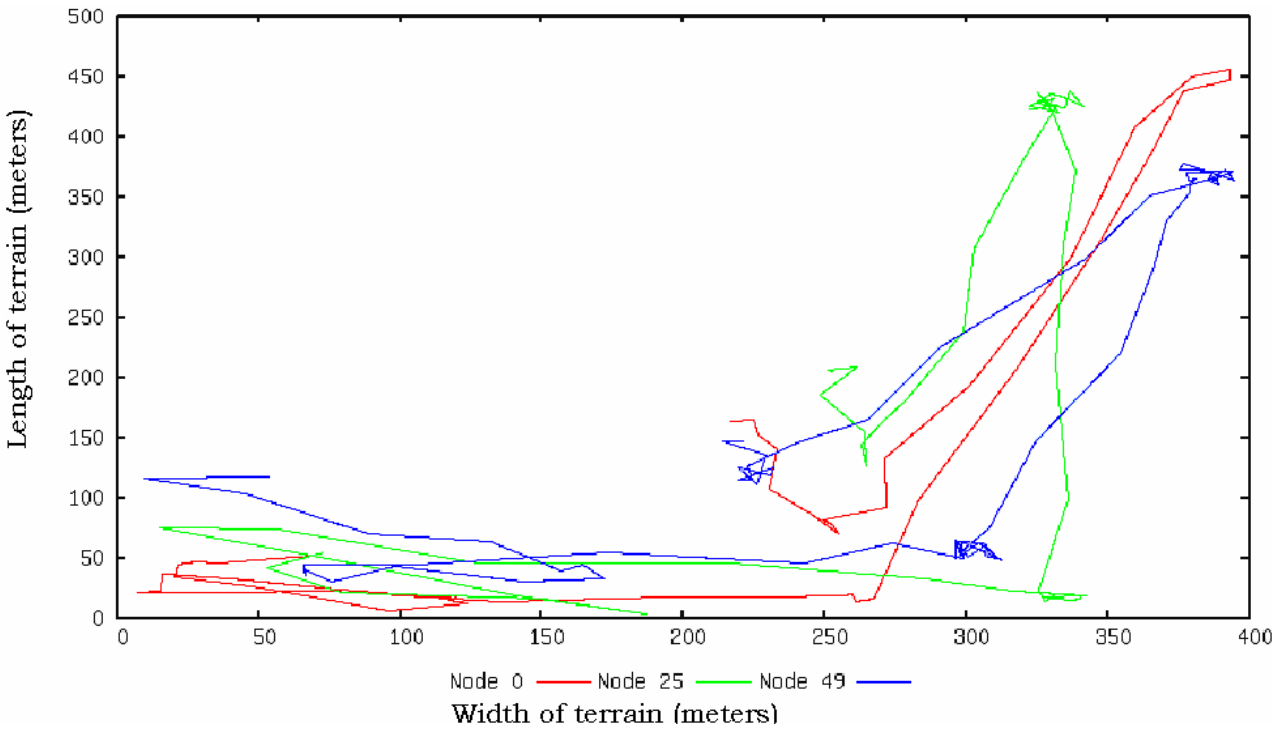

Fig.5.3. Travelling pattern of 3 mobile nodes with RPGM model

5.2.4 Community based mobility model

The authors in [10] tried to model the mobility pattern of mobile nodes based on the social relationship of humans that carry the devices. To that end they propose a mobility model based on social network theory. The model allows collections of hosts to be grouped together in a way that is based on social relationships among the individuals. This grouping is then mapped to a topographical space, based on the strength of social relation.The movements of the hosts are also driven by the social relationships among them. The model also allows for the definition of different types of relationships during a certain period of time (i.e., a day or a week).

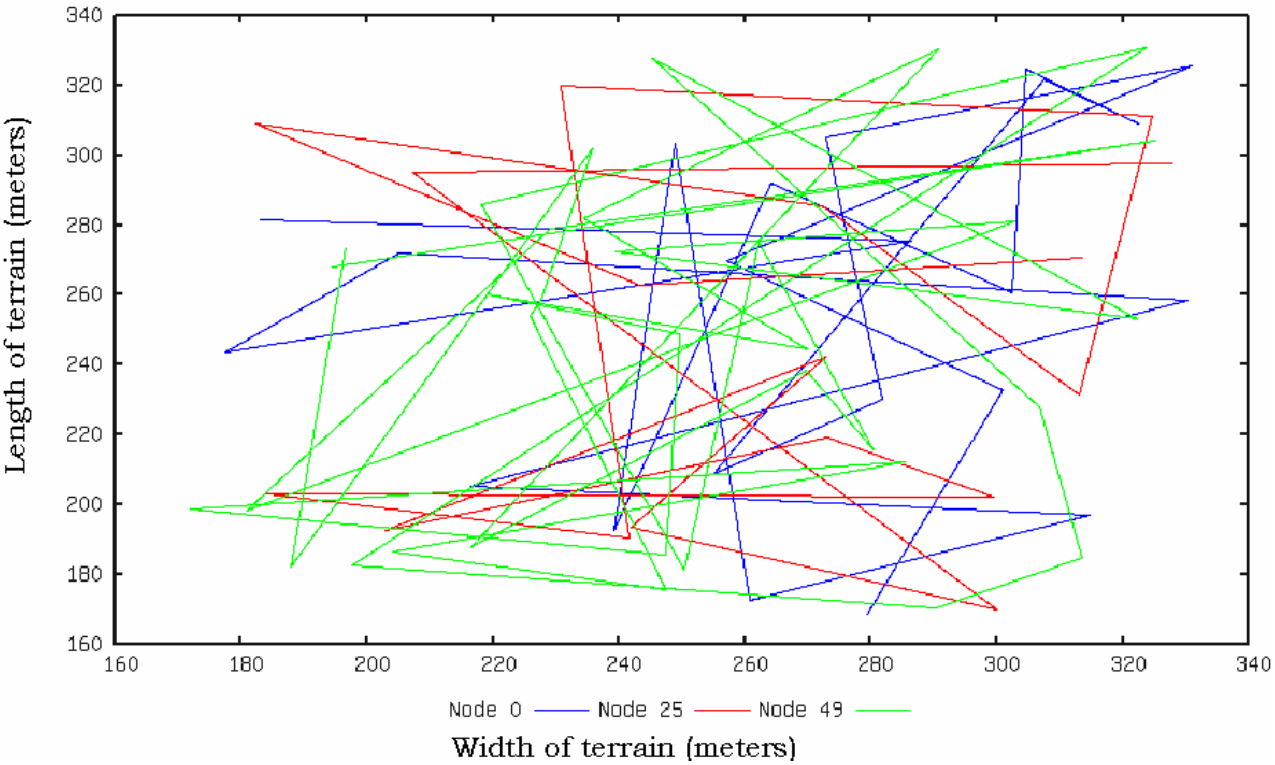

Fig.5.4.Traveling pattern of 3 mobile nodes with Community based mobility model

The screen capture in Figure 5.4 illustrates the travelling pattern of three mobile nodes among fifty nodes following the community based mobility model. One group of mobile nodes was initially chosen and the topology area was divided into 3 rows and columns. The speed of travellers was set at 3.0 m/s.

The travelling patterns demonstrated above which are generated in this thesis work as well as all subsequent analyses carried out using the random walk, random waypoint, and RPGM models are based on packages obtained from researchers at Toilers group from Colorado school of Mines [9]. The packages are obtained upon request by e-mail. The pattern for the community based mobility model was generated using the mobility implementation that can be downloaded from the following site which is available as public domain as quoted in [10].
http://ww.cs.ucl.ac.uk/staff/m.musolesi/mobilitymodels

5.2 Implementing PACL-SDP

Adding a new implementation into the NS 2 package involves several steps. The implementation of the new protocol PACL-SDP has two parts. The first one resides at the application layer and the second at the transport layer. There have been also modifications carried out at the routing layer protocol, AODV, to give it service discovery capabilities. Consider Figure 5.5.

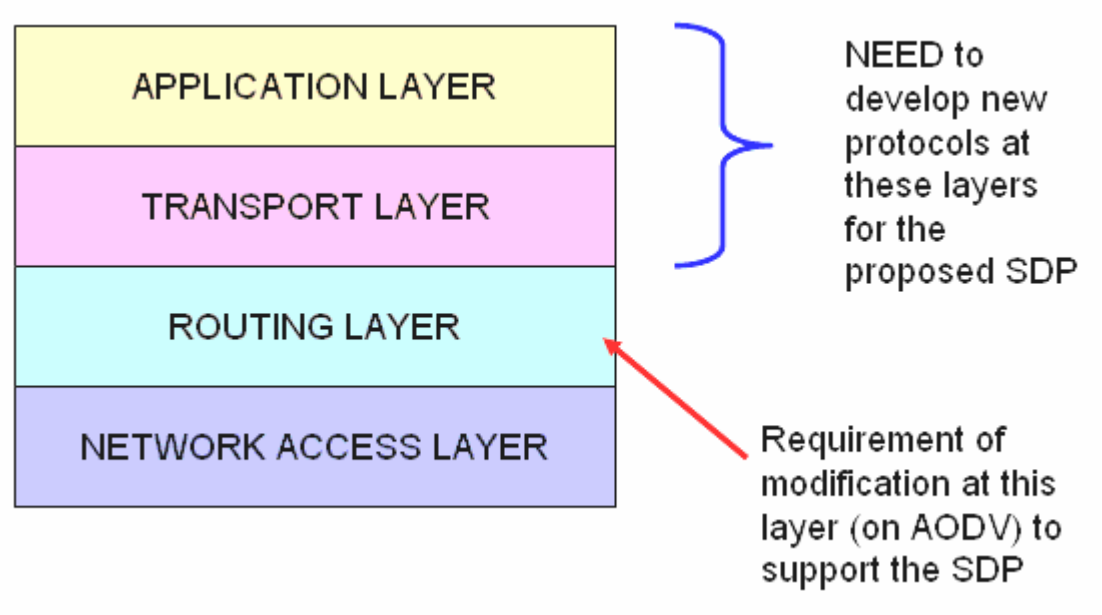

Fig. 5.5 The traditional TCP/IP network model

Protocol implementations in NS 2 consist of C++ files. These files are usually in pairs, the header file ".h file" containing class declarations and variable definitions, and the source code file ".cc file" containing the actual implementations of processes (or functions). The development of PACL-SDP is no exception. To that end the package developed for this new cross-layered protocol consists of LS_App.h and LS_App.cc files at the application layer, and LS_Agent.h and LS_Agent.cc files at the transport layer. The newly introduced packets, the leader selection packet (LS_Pkt) and the service discovery packet (SERVICE_Pkt) were the other additions into the NS 2 package.

In addition to the above newly created files, there are several NS 2 files that need to be adjusted to integrate these new files into the package and run the simulation seamlessly. These files include *aodv.h/.cc*, *aodv_packet.h, packet.h, ns-default.tcl, ns-packet.tcl, trace.h, cmu-trace.h/.cc, Makefile, install, and Makefile.in*. The modification with the last two files was carried out for the purpose of enabling GDB (GNU debugger) with NS 2.

5.2.1 Leader Election

A node that has the richest computing resource among a group (or cluster) of nodes becomes the head of the cluster. In the election process, the first node aware of the absence of a leader or any cluster formation, generates a leader selection packet (LS_Pkt) and broadcasts it to its neighbours after recording its processing capacity. The neighbours participate in the process by responding to the received packet based on the criteria they posses. The detail of the process is described in section 4.2.

In the implementation of the process, each node is given a random timer upon instantiation (or when the nodes are created). The node whose random timer expires first becomes the INITIATOR.

The other nodes, once they received an LS_Pkt from an INITIATOR node, cancel their timer and start processing the received packet. These nodes also record the address of the INITIATOR from the ORIGINATOR_IP field of the LS packet. If a non-INITIATOR node receives another broadcast LS_Pkt coming from a neighbouring cluster or from members of its cluster, it drops the packet as it had already received one. On the other hand, if the incoming packet is a unicast packet, the recipient checks the ORIGINATOR_IP field of the packet. If the address in the ORIGINATOR_IP field is not the same as its recorded address for the INITIATOR, it discards the packet. On the other hand, if the addresses are the same, the receiving node checks the WINNER FLAG. The WINNER FLAG is set to declare that this node has won the leader election process.

Note that in the simulation the processing capacities in terms of memory size and processor speed are also assigned randomly to each node during the creation of the nodes. The code lines here below show how the random waiting time and the memory size are set for each node.

```
wait_pd = Random::uniform(100.0)+13.4 // waiting time before generating
                // LS_Pkt . 13.4 is added to avoid transient effects.

my_memory = int(Random::uniform(3936) + 64); // setting memory randomly
                                              //in MB
```

The output of a trace file image shown in Figure 5.6 illustrates that an INITIATOR (node 54) has sent a unicast message to a node (node 71) notifying it as the leader of the cluster group. The lower part of the figure shows how node 71 processes the received LS_Pkt with the WINNER FLAG set. Note from the underlined field that the packet type is LS (Leader selection).

The output trace file for the discussions used throughout this section is based on simulation results obtained for the RPGM model.

From Figure 5.6 it can be seen that node 54 started the leader election process for this particular cluster at around t =16.37sec. At that period and during the exchange of the LS_Pkt the WINNER FLAG was set at zero. (see entry –Pw 0). “w” stands for WINNER_FLAG. Hundred seconds later (which is the timeout period set for an INITIATOR to collect LS_Pkts), the INITIATOR stops receiving any more LS_Pkts and notifies the winner based on the received packets thus far. To that end it sets the WINNER FLAG to one, and unicasts the message to the winner node. The hundred seconds waiting time period is an arbitrary value. Otherwise, it can be adjusted based on some experimental values.

After receiving the winner notifying packet (in this case node 71), the winner node resets the flag and passes a message to its lower layer for the inclusion of leader announcement information in the routing layer broadcast messages. This is shown at the bottom part of the figure. The underlined field indicates that the packet type is LS (Leader Selection) shown earlier in Figure 4.1.

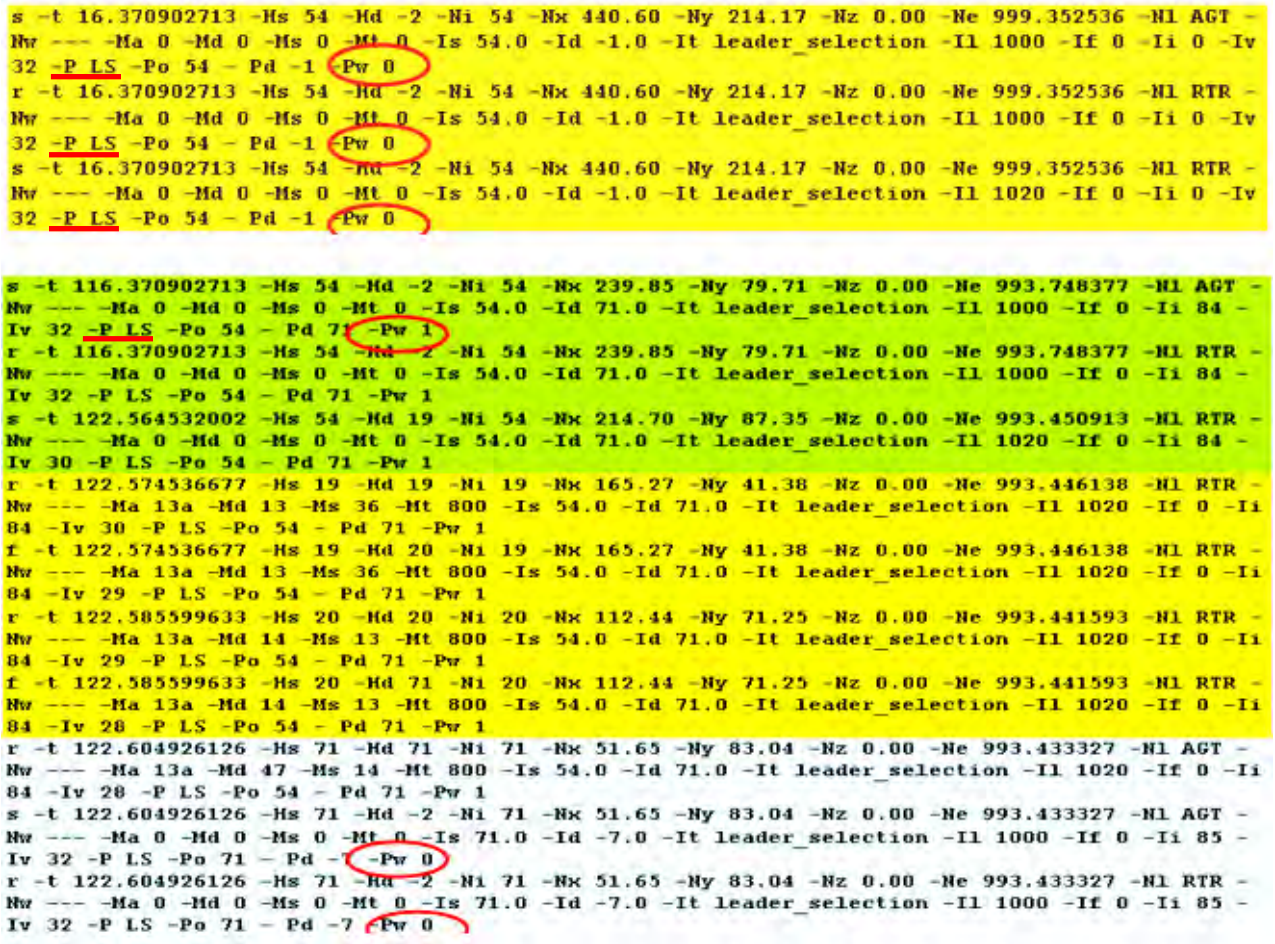

Fig. 5.6 Trace-level leader election process

5.2.2 Service registration

A node possessing a service to offer has to register its services at two locations. The first is at a local service cache and the second is at the broker node. To carryout these, the SERVICE_Pkt has dedicated service IDs to differentiate among the different service discovery processes. Note that the service ID (S_ID) here is the equivalent of "Function ID" from SLP v.2 [29].

The tcl script command shown below instantiates a service at the application layer of a node.

```
$ns_ at 205.0 "$ls_app_(58) set-service PRINT"

# where ls_app_(58) the service discovery application layer process for
node 58. Now this node has installed a printer service.
```

Next, the node registers that service at its local cache. Following that periodically it registers the services by sending service binding registration packets to the directory node. For this thesis work a server is assumed to possess one type of service. Observe from the underlined field that the packet type is SERVICE one that was depicted in Figure 4.7.

The screen capture of a trace file output shown in Figure 5.7 depicts two nodes, (node 58) and (node 81), which are recording their services at their local caches. The registration at the directory node is carried out similarly with PS_ID set at 5 and the destination address set at the address of the directory node.

```
s -t 205.000000000 -Hs 58 -Hd -2 -Ni 58 -Nx 155.58 -Ny 91.12 -Nz 0.00 -Ne 990.135813 -Nl AGT
-Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 58.0 -Id -7.0 -It node_service -Il 1000 -If 0 -Ii 89 -Iv
32 -P SERVICE -Po 58 - Pd -7 -PS_ID 1
r -t 205.000000000 -Hs 58 -Hd -2 -Ni 58 -Nx 155.58 -Ny 91.12 -Nz 0.00 -Ne 990.135813 -Nl RTR
-Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 58.0 -Id -7.0 -It node_service -Il 1000 -If 0 -Ii 89 -Iv
32 -P SERVICE -Po 58 - Pd -7 -PS_ID 1

s -t 207.200000000 -Hs 81 -Hd -2 -Ni 81 -Nx 64.50 -Ny 44.67 -Nz 0.00 -Ne 990.050788 -Nl AGT
-Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 81.0 -Id -7.0 -It node_service -Il 1000 -If 0 -Ii 94 -Iv
32 -P SERVICE -Po 81 - Pd -7 -PS_ID 1
r -t 207.200000000 -Hs 81 -Hd -2 -Ni 81 -Nx 64.50 -Ny 44.67 -Nz 0.00 -Ne 990.050788 -Nl RTR
-Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 81.0 -Id -7.0 -It node_service -Il 1000 -If 0 -Ii 94 -Iv
32 -P SERVICE -Po 81 - Pd -7 -PS_ID 1
```

Fig. 5.7 Service registration at local cache

The data structure used to define the contents of a service cache is given here below.

```
struct service_item         // Defines the contents of the service cache
  {
    int service_code;
    nsaddr_t server_IP;
    double service_time;
    service_item* nxt;    // A pointer to service_item
  };


struct service_cache         // Defines the service cache in the memory
 {
   service_item* head;
 };
```

Nodes follow the routine given here to store service bindings into their cache.

```
void AODV::push(service_cache& sc, service_item* item)
 {
     if(is_empty_cache(sc))  // If cache is empty put it as first record
       {
        sc.head = item;
       }
 else
 {
    service_item* found =NULL;
    found = search_addr(sc, item->server_IP); // search if there is an old entry

    if(found)                          // If so, then refresh entry point
     {
      found->service_time = item->service_time;
     }
    else               //otherwise, enter the record on top of the stack
      {
        item->nxt = sc.head;
        sc.head = item;
      }
  }

 }
```

The subroutine used in the storing process,

search_addr(service_cache&, nsaddr_t)

is defined as illustrated below.

```
service_item* AODV:: search_addr(service_cache sc, nsaddr_t addr)
 {
       service_item* found = sc.head;
       while(found)
         {
           if(found->server_IP == addr)
               return found;
           else
              found = found->nxt;
         }
       return NULL;
 }
```

5.2.3 Service request

A client node seeking a certain type of service issues a service request from the upper layer and passes it to the lower (routing) layer.

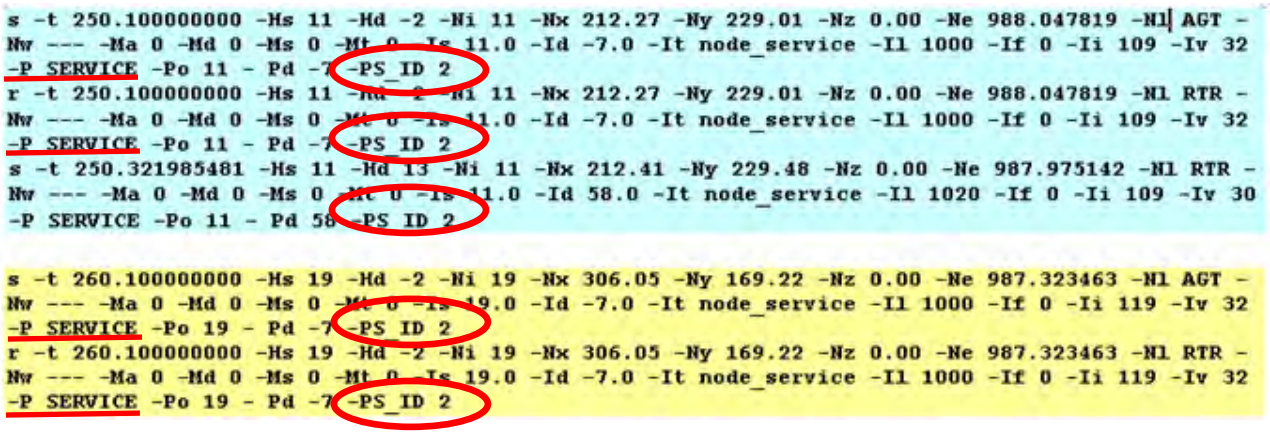

Fig. 5.8 Service request process

The trace output capture of Figure 5.8 illustrates that two nodes, node 11 and node 19 have passed service request messages to their lower layers. Notice that the destination address of the packet is set to "-7" as the request is passed to the lower layer.

This is to indicate that the packet is meant to the lower layer and not to any node in the network. The application layer doesn't know who the possible provider of the address of the server is. It just sends the request to its lower layer.

The routing layer consults the neighbours' cache for service binding. If a binding is found, the resolved address will be used to send the request back to the upper layer. If not, the service binding request will be directed to the directory node. The directory node then sends back a service binding reply if a service provider is found.

Figure 5.9 below illustrates that from the two nodes that requested for a service, node 19 could not find the service in its neighbours. Hence, it has to generate a service binding request to its leader node, node 71. The upper part of the figure shows this step.

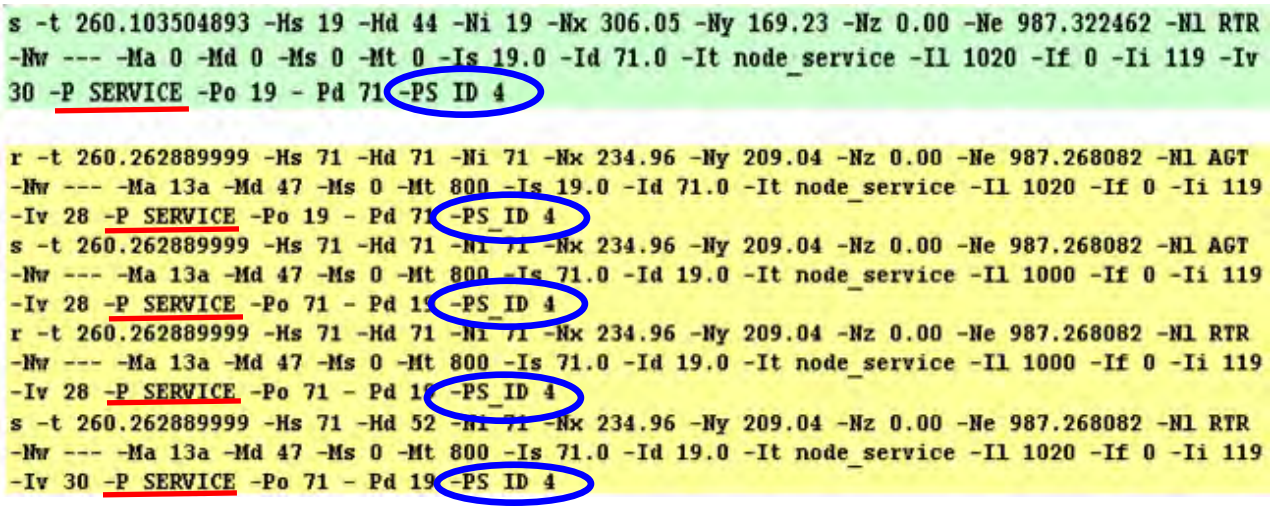

Fig. 5.9 Service binding request reply

The lower part of the figure shows that leader node 71 has received a binding request, has resolved the address and has sent service binding reply back to the client node, node 19.

The directory chooses the server with the highest service lifetime if there exist two or more servers providing the same service. The search facility employed by the directory node is as illustrated in the piece of code shown below. In fact, the same facility is used by member nodes when looking for bindings in neighbours' cache.

```
service_item* AODV::search_service_code(service_cache sc, int code)
 {
    service_item* found = sc.head;
    service_item* record = NULL;
    double life_time=0;

    while(found)
      {
          if(found->service_code ==code)
            {
               if(!life_time)        // The first server node identified
                {
                 record=found;
                 life_time= found->service_time;
                 found=found->nxt;
                }
              else
               {
                  if(life_time < found->service_time)
                    {
                      record = found;
                      life_time = found->service_time;
                      found=found->nxt;
                    }
                  else
                     found=found->nxt;
               }
            }
          else
            found = found->nxt;
      }

    return record;
 }
```

After receiving the service binding reply, node 19 now re-send its service request to the resolved server, here node 81. The screen capture in Figure 5.10 shows that step.

```
s -t 260.282618898 -Hs 19 -Hd 19 -Ni 19 -Nx 306.35 -Ny 169.48 -Nz 0.00 -Ne 987.262193 -Nl AGT -
Nw --- -Ma 13a -Md 13 -Ms 34 -Mt 800 -Is 19.0 -Id 81.0 -It node_service -Il 1000 -If 0 -Ii 119
-Iv 29 -P SERVICE -Po 19 - Pd 81 -PS_ID 2
r -t 260.282618898 -Hs 19 -Hd 19 -Ni 19 -Nx 306.35 -Ny 169.48 -Nz 0.00 -Ne 987.262193 -Nl RTR -
Nw --- -Ma 13a -Md 13 -Ms 34 -Mt 800 -Is 19.0 -Id 81.0 -It node_service -Il 1000 -If 0 -Ii 119
-Iv 29 -P SERVICE -Po 19 - Pd 81 -PS_ID 2
```

Fig. 5.10 Sending service request to a resolved server

5.2.4 Service reply

In this thesis work the service reply packet is used to demonstrate that the server can successfully contact the client. It is analogous to the first message delivered to the client from the server host. Otherwise, how services are invoked is not the scope of the thesis.

```
s -t 250.438131504 -Hs 58 -Hd 58 -Ni 58 -Nx 226.34 -Ny 160.96 -Nz 0.00 -Ne 987.933987 -Nl AGT
-Nw --- -Ma 13a -Md 3a -Ms 41 -Mt 800 -Is 58.0 -Id 11.0 -It node_service -Il 1000 -If 0 -Ii
109 -Iv 28 -P SERVICE -Po 58 - Pd 11 -PS_ID 3
r -t 250.438131504 -Hs 58 -Hd 58 -Ni 58 -Nx 226.34 -Ny 160.96 -Nz 0.00 -Ne 987.933987 -Nl RTR
-Nw --- -Ma 13a -Md 3a -Ms 41 -Mt 800 -Is 58.0 -Id 11.0 -It node_service -Il 1000 -If 0 -Ii
109 -Iv 28 -P SERVICE -Po 58 - Pd 11 -PS_ID 3
s -t 250.438131504 -Hs 58 -Hd 65 -Ni 58 -Nx 226.34 -Ny 160.96 -Nz 0.00 -Ne 987.933987 -Nl RTR
-Nw --- -Ma 13a -Md 3a -Ms 41 -Mt 800 -Is 58.0 -Id 11.0 -It node_service -Il 1020 -If 0 -Ii
109 -Iv 30 -P SERVICE -Po 58 - Pd 11 -PS_ID 3

r -t 250.471191057 -Hs 11 -Hd 11 -Ni 11 -Nx 212.51 -Ny 229.79 -Nz 0.00 -Ne 987.923956 -Nl AGT
-Nw --- -Ma 13a -Md b -Ms d -Mt 800 -Is 58.0 -Id 11.0 -It node_service -Il 1020 -If 0 -Ii 109
-Iv 28 -P SERVICE -Po 58 - Pd 11 -PS_ID 3
```

(a)

```
s -t 260.402121262 -Hs 81 -Hd 81 -Ni 81 -Nx 151.02 -Ny 122.95 -Nz 0.00 -Ne 987.220141 -Nl AGT
-Nw --- -Ma 13a -Md 51 -Ms 12 -Mt 800 -Is 81.0 -Id 19.0 -It node_service -Il 1000 -If 0 -Ii
119 -Iv 28 -P SERVICE -Po 81 - Pd 19 -PS_ID 3
r -t 260.402121262 -Hs 81 -Hd 81 -Ni 81 -Nx 151.02 -Ny 122.95 -Nz 0.00 -Ne 987.220141 -Nl RTR
-Nw --- -Ma 13a -Md 51 -Ms 12 -Mt 800 -Is 81.0 -Id 19.0 -It node_service -Il 1000 -If 0 -Ii
119 -Iv 28 -P SERVICE -Po 81 - Pd 19 -PS_ID 3
s -t 260.402121262 -Hs 81 -Hd 18 -Ni 81 -Nx 151.02 -Ny 122.95 -Nz 0.00 -Ne 987.220141 -Nl RTR
-Nw --- -Ma 13a -Md 51 -Ms 12 -Mt 800 -Is 81.0 -Id 19.0 -It node_service -Il 1020 -If 0 -Ii
119 -Iv 30 -P SERVICE -Po 81 - Pd 19 -PS_ID 3

r -t 260.434042327 -Hs 19 -Hd 19 -Ni 19 -Nx 306.61 -Ny 169.70 -Nz 0.00 -Ne 987.212781 -Nl AGT
-Nw --- -Ma 13a -Md 13 -Ms 32 -Mt 800 -Is 81.0 -Id 19.0 -It node_service -Il 1020 -If 0 -Ii
119 -Iv 28 -P SERVICE -Po 81 - Pd 19 -PS_ID 3
```

(b)

Fig. 5.11 Process of service reply (a) for node 11, (b) for node 19

Trace file output shown in Figure 5.11 (a) above demonstrate that server node 58 has sent a service reply to its client node 11. The lower part of the same figure shows that the client has successfully received the reply. The capture in Figure 5.11 (b) shows the similar process for client node 19 and server node 81.

5.3 Evaluation Metrics

The evaluation metrics used in the research work are of two types. The first group of metrics is used to indicate the amount of discovery overhead traffic incurred by the network. These are useful to estimate what is referred to as the communication cost. The next group of metrics used is evaluation parameters that indicate the performance of the network in terms of successful delivery of packets, the response time required in the process, and consumption of energy. In addition, the impact of the service discovery process on throughput, packet delivery ratio, and end-to-end delay for CBR (Constant bit rate) packets are analyzed.

5.3.1 Measuring Communication cost

The cost of communication is a measure of overhead traffic that is employed by a protocol to facilitate successful transmission of actual data traffic. The cost could be in terms of unicast and/or broadcast control messages. In service discovery protocols there are unicast overhead messages in those protocols that employ active discovery method. In such situations, server nodes periodically update their offered services to directory nodes by unicasting service binding information. In addition, client nodes seeking services can unicast service binding requests to directory hosts, and the directory hosts can in turn send back unicast service binding messages. In this architecture, directory nodes periodically announce their presence by flooding directory information into the network. This is a broadcast overhead message.

On the other hand, for architectures that employ passive discovery approach, there can be broadcast messages coming from several servers advertising their services into the network.

For this research work, the following overhead messages have been identified and are used in the comparative analysis.

i) Unicast service binding registration messages to a directory node
ii) Unicast service binding request messages to a directory node
iii) Unicast service binding reply messages from a directory node
iv) Broadcast directory announcing messages

5.3.2 Measuring Network Performance

The following network performance metrics have been used for analysis in this work.

i) Average energy consumption of a node

- For the traditional and cross-layered protocols considered, the amount of energy a node is left with (the remaining energy) after successfully discovering a sought service is used as a measure of efficient energy utilization by a protocol. Each node has been supplied with initial energy of 1000 Joules.

ii) Average response time

- The amount of time elapsed from the request of a service by a client node until successful reception of service reply from a server is used as a measure of time utilization of a protocol.

iii) Successful service discovery (SSD)

- This metric is used to measure the success rate of a protocol in discovering or locating servers that provide requested services. It is the ratio of successfully discovered services to the total number of service request messages.

In carrying out the simulation to compute the above seven metrics, client nodes identified for a selected cluster were organized into several groups of five clients each. Without loss of generality, the service request process was carried out with the first group of five clients sending their messages at close intervals. On the next round, another group of five clients would be included after about ten seconds for the group mobility models or after five seconds for the entity mobility models keeping the first group active. On the third round, another group of five clients would join the first two groups in a similar manner and send their request messages. The process continues until all the groups take part in the request-response process. The step is as shown in table 5.1 at the last row. A sample code depicting how nodes issue service request is shown in Appendix A. Appendix B gives sample commands used for data gathering and analyses. This procedure is useful to show how the network handles the gradual increase of the workload (number of service request messages).

iv) Packet delivery ratio (PDR)

This is a measure of the number of successfully received packets to the total number of packets sent. The traffic source used in the study is a CBR source.

v) Average Throughput

This is the measure of the number of packets/bytes/bits successfully received by a node per unit of time. The traffic source employed here too is a CBR source

vi) Average end-to-end delay

This is the sum of delays incurred due to buffering during route discovery, queuing delay for different processes, transmission delay at MAC layer, and propagation delays. The computed values for this metric are based on a CBR traffic sources and corresponding client nodes.

In computing the above three metrics, five CBR sources were chosen from the existing members of the cluster. Five other nodes were chosen to serve as clients for the five CBR sources. Next, the CBR sources generate packets of 1024 bytes for hundred seconds to their clients. The data rates for the CBR sources were varied and the procedure was repeated. Each repetition was conducted while the other members of the cluster are still active in their service request-response process. This step shows the effect of the service discovery processes on data transfer in a network with several actively communicating nodes. After the end of the simulation, the PDR for the client nodes, average throughput of the five CBR clients, and the average end-to-end delay of CBR packets received by the clients were computed.

5.4 Analysis of Results

The results of the research and the comparative analyses of the cross-layered protocol with that of the traditional directory based service discovery protocol, SLP_MANET, for the aforementioned metrics is presented in this section. The application layer based protocol was implemented so that both protocols can be compared for performance on the same network. Moreover, it is a requirement that both protocols should possess similar features on parameters that is common to both. One such parameter is for example the time period for which a server initiates a registration message to a directory node.

Tables 5.1 and 5.2 below summarize the network setup used in carrying out the simulation study. According to [11] factors that can affect service availability in mobile ad hoc networks include:

- Server and client density
- Frequency of advertisement by service coordinators
- Frequency of service registration by servers
- Service querry search radius
- Rate of mobility
- Number of service requests

Table 5.1 presents the general setup used in the study taking into consideration the aforementioned factors. Table 5.2 gives the parameter setups used for the CBR traffic generation that is used for computation of throughput, packet delivery ratio, and end-to-end delay.

The results presented in the graphs that follow as well as subsequent discussions based on outcomes obtained for the default seed. Otherwise, for the purpose of verifying the model, four additional runs are made at different seed values for the Random Waypoint mobility model and presented at the end of the section. This helps to see that conclusions drawn from the analyses are independent of the selection of seed values.

Table 5.1 summary of overall simulation setup

Network parameters	**Mobility Model**			
	Community based	**RPGM**	**Random Walk**	**Random Waypoint**
Topology area	500m X 500m	500m X 500m	500m X 500m	500m X 500m
Simulation time	500.0 sec	500.0 sec	500.0 sec	500.0 sec
Radio range	40m	80m	200m	200m
Number of nodes in the network	100	100	140	140
Number of nodes in the selected cluster	54	79	91	132
Average speed	3.0 m/s	3.5m/s	3.5 m/s	3.5 m/s
Speed delta	-	1.0 m/s	1.0 m/s	1.0 m/s
Pause time	-	20.0 sec	-	20.0 sec
Pause delta	-	5.0 sec	-	5.0 sec
Number of servers	5	10	15	20
Number of client requests	5,10,15,20,25, 30,35,40,45,50	5,10,15,20,25, 30,35,40,45,50 55,60,65,70,75	5,10,15,20,25, 30,35,40,45,50 55,60,65,70,75,80	5,10,15,20,25, 30,35,40,45,50 55,60,65,70,75, 80,85,80,95,100

Table 5.2 summary of CBR traffic setup

Traffic type used	CBR
Packet size	1024 bytes
Data rate	10,40,70,100,140, 170,200 Kbps
Transmission duration	100.00 sec
No. of CBR sources	5
No. of CBR clients	5

5.4.1 Comparative analysis of Network Communication cost

A) Unicast overhead messages.

i) Unicast service binding registration messages

Figure 5.12 demonstrates the comparison of the unicast messages incurred into the network during service registration by different number of servers. Normally, for both protocols there can not be a significant difference in the number of messages in this category. Nonetheless, as the ways the registration messages are processed is different for the two protocols, there exists a slight difference in the number of messages generated. Note that what has been illustrated by the figure is the total sum of messages generated by all the servers within the cluster. Otherwise, on a per server basis the traditional protocol has only one more additional message than its cross-layered equivalent. For example, for the RPGM model one of the servers was observed to have updated its registration 13 times under SLP_MANET protocol, while the same server had updated the same service registration 12 times under PACL-SDP. Another server was observed to register 17 times and 16 times under protocols SLP_MANET and PACL-SDP, respectively. The situation is no different for the other servers. Hence, when the total sum of registrations made by all servers is considered (for a total of 10 servers) the value for SLP_MANET is ten more than that value for PACL-SDP.

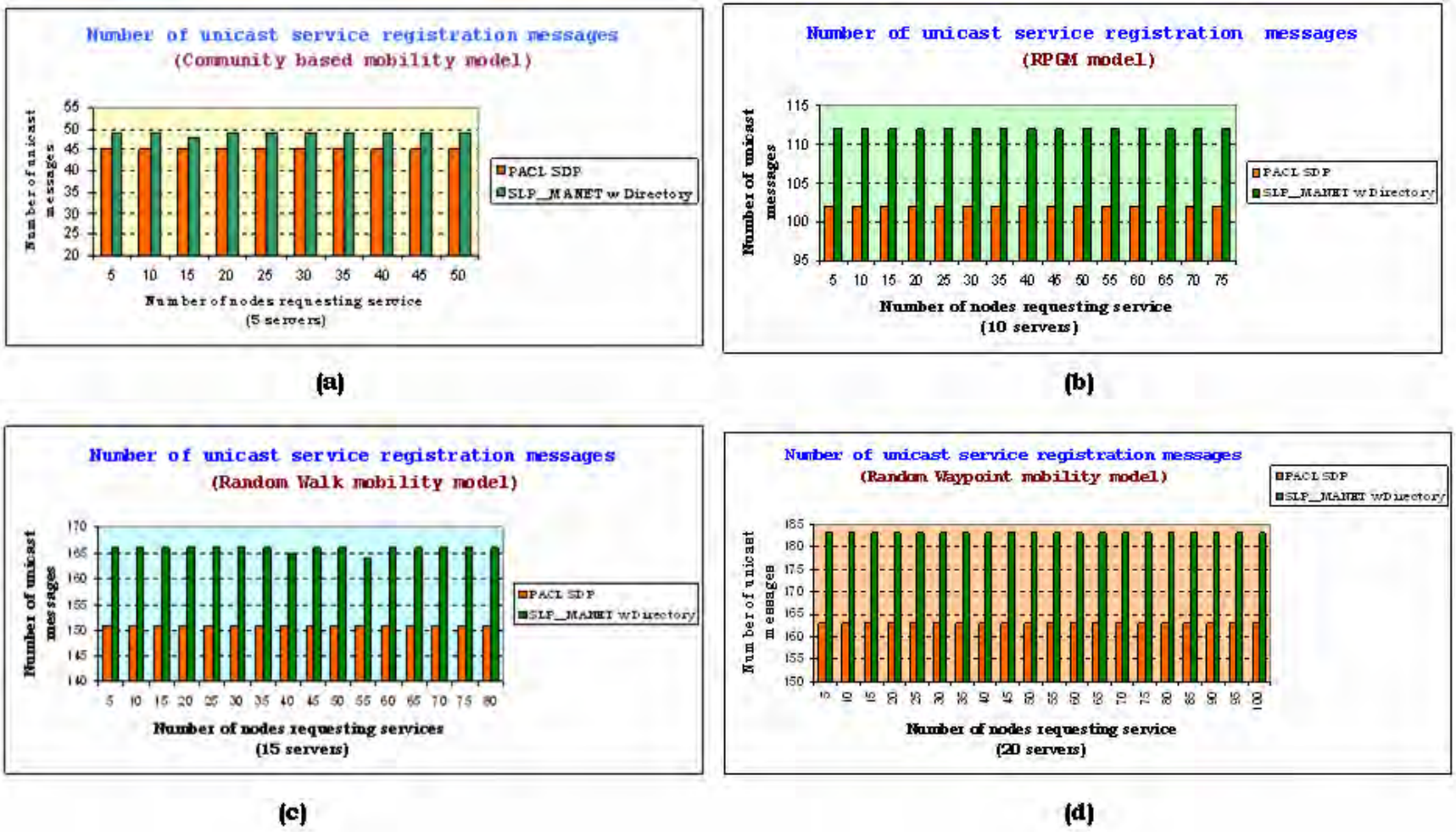

Fig. 5.12 Comparison of number of unicast registration messages

Similarly, when Random Waypoint is considered (here there are 20 servers), the total sum of registrations made by all servers for the traditional protocol is 20 more than the value for the cross-layered protocol. As can be inferred from the figure, the same thing happens for the other mobility models too. The reason behind can be explained as follows. In both protocols, the first initiation for registration takes place at the upper layer. For the traditional protocol it is always this layer that carries out the periodic registration procedure. On the other hand, for the cross-layered protocol, the periodic registration step is delegated to the routing layer. Hence, the cross-layered protocol can be somewhat late to issue its first registration message. After this initial message the periodic registration message is the same as that of the traditional protocol. However, the cross-layered protocol has one less message at the end of the simulation period.

ii) Unicast service binding request messages

Fig. 5.13 illustrates the unicast messages for service binding request messages. Clearly, as a node employing the upper layer based protocol is not aware of services in its vicinity, it is always relying on the directory node for service discovery. Hence, regardless of the mobility model used, the upper layer based protocol incurs more unicast messages. On the other hand, a node using the cross-layered protocol does not need to generate a binding request to the directory node if it can discover a service from its neighbours.

iii) Unicast service binding reply messages

This is a direct consequence of the above type of message. If there are more service binding requests made to a directory node in the network, then there can be more binding replies in return coming from the directory node. Consequently, the traditional protocol has always more binding reply messages than the cross-layered equivalent. This is depicted in Figure 5.14.

It is to be noted that the directory node may not receive all binding requests coming from the client nodes due to collisions. Consequently, binding request replies coming from the directory as shown in Figure 5.14 may be less than the number of binding requests coming into the network from the clients as depicted in Figure 5.13.

Figure 5.15 depicts the comparison for the total number of unicast messages introduced by the two protocol types. It is the sum of the registration messages made by server nodes, binding request messages made by client nodes, and binding reply messages coming from the directory node. In addition, not all client nodes may be able to receive It can in general be observed that the traditional protocol incurs more unicast overhead messages than the cross-layered protocol.

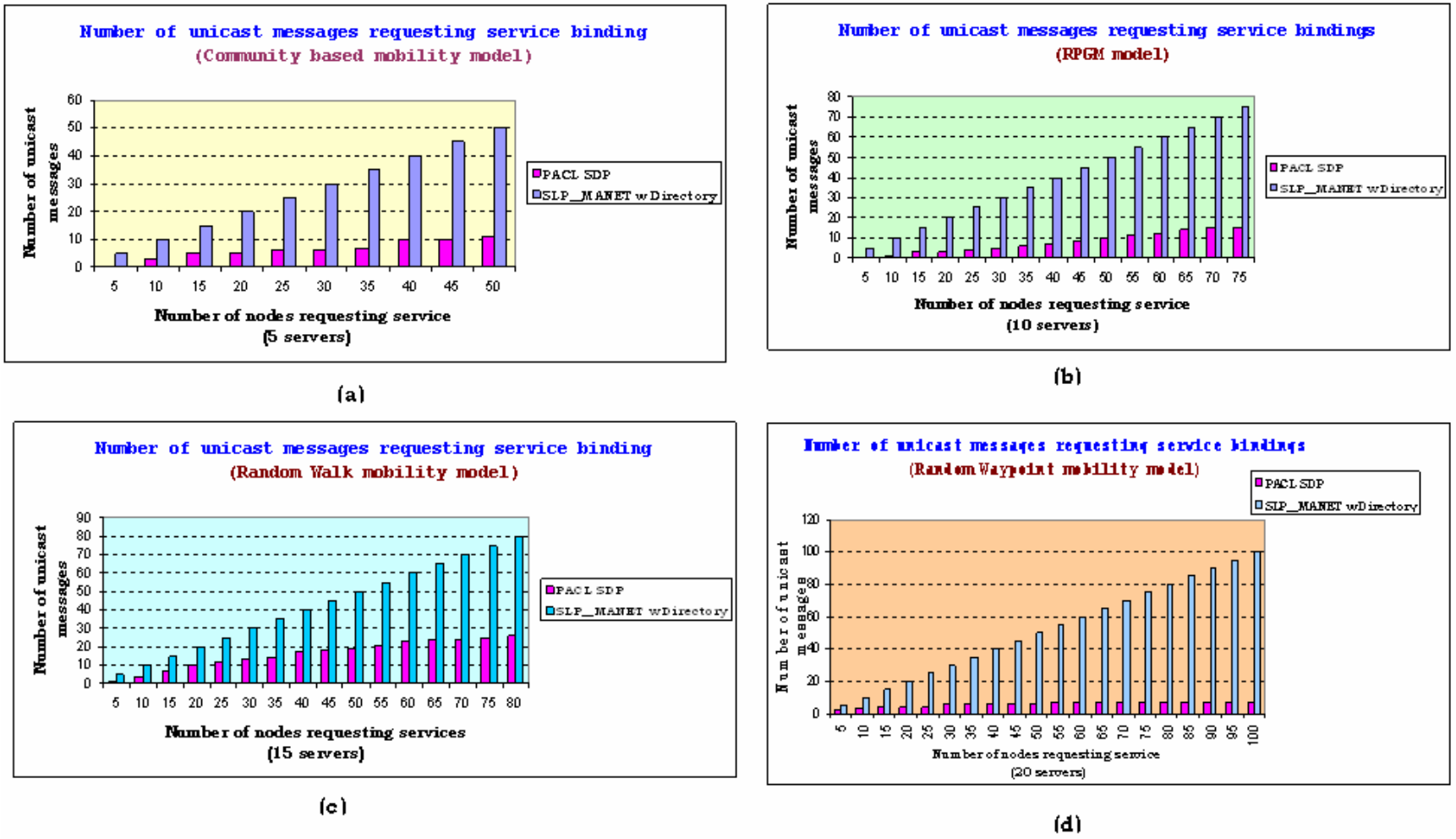

Fig. 5.13 Comparison of number of unicast service binding request messages

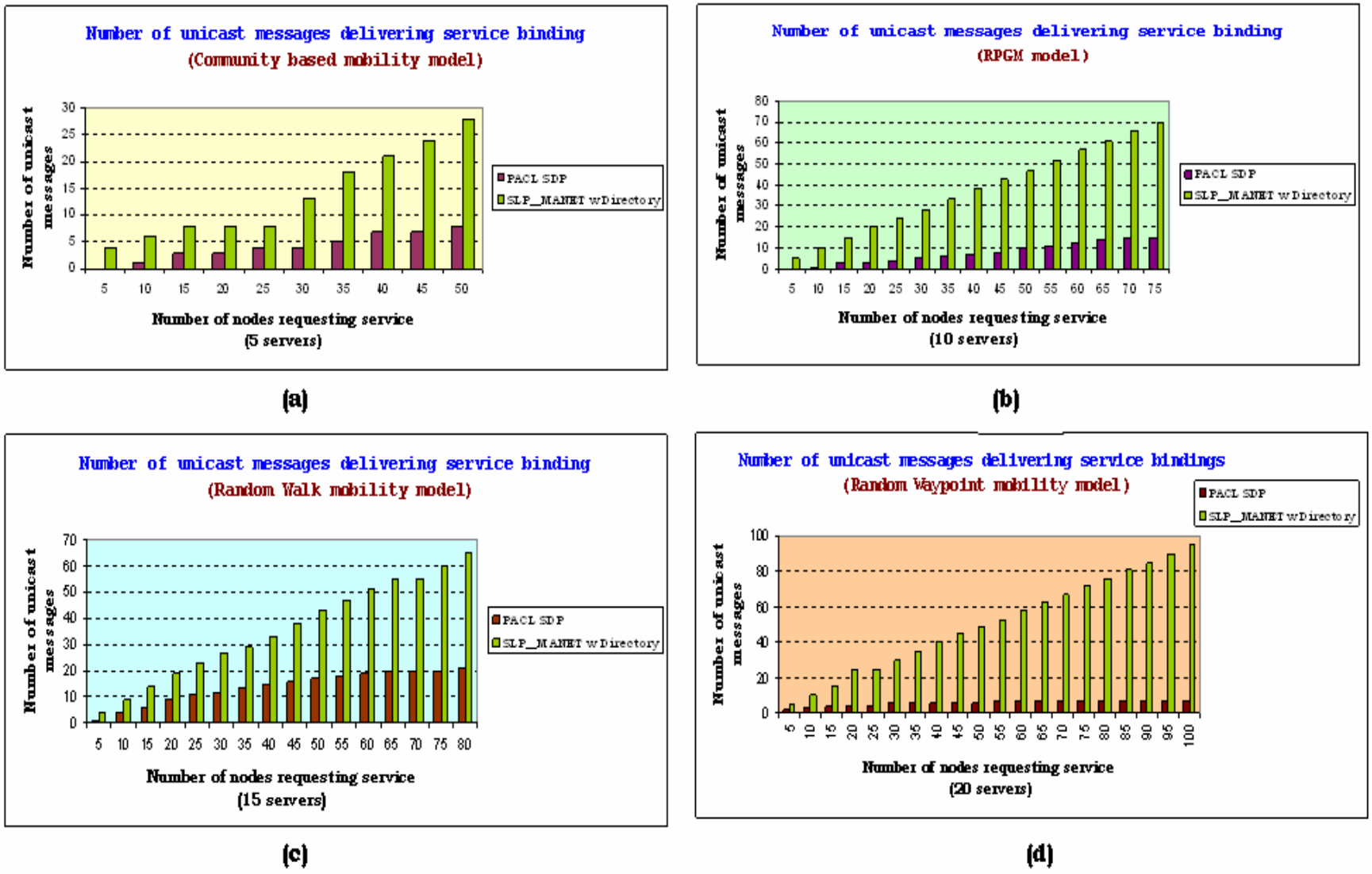

Fig. 5.14 Comparison of number of unicast service binding reply messages

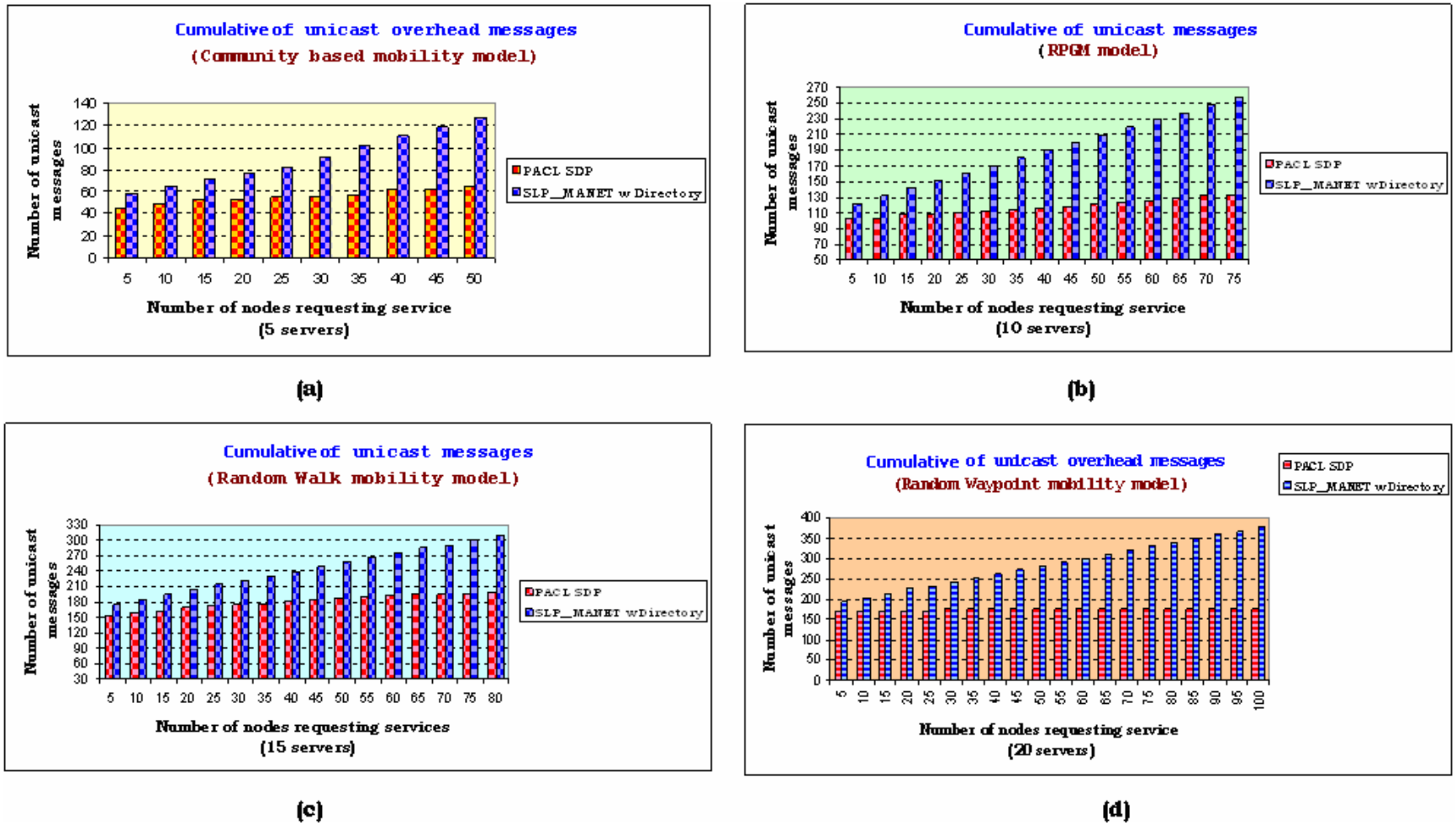

Fig. 5.15 Comparison of number of total unicast messages

B) Broadcast overhead messages

The number of broadcast messages introduced into the whole network by the traditional and cross-layered algorithms is shown in Figure 5.16. Note that the figure shows the broadcast messages throughout the whole network and not just the selected cluster. The contributors to the broadcast messages are the leader election process and the periodic advertisement by the directory node. While the first type of broadcast message is a feature pertinent for the whole network nodes involved in the leader election process, the latter is a feature available only for the considered cluster. This is because service related characteristics are defined only for the selected cluster only. Hence, it can be seen that the broadcast incurred by the network due to periodic advertisement of the directory node is much more significant than the leader election process. This is so because flooding messages due to the leader election process are over once a leader is elected. On the other hand, for the traditional protocol the elected leader is engaged in periodic announcement of its leadership. In the simulation setup used in this study, the directory node announces its leadership every 30.0 seconds. The other nodes declare the absence of a leader node if no leader node announcing message is received in 60 seconds after the last update. Note that the periodic directory announcement is a feature for the traditional protocol only. Otherwise, the cross-layered protocol does rely on already available routing layer messages to piggyback its advertisement. Hence, it does not suffer from this type of overhead message.

Figure 5.17 presents the cumulative sum of all types of overhead messages introduced into the network by the two protocols. The proportion of these overhead messages when the given cluster has its entire client nodes fully active in service discovery processes are illustrated in Figures 5.18 and 5.19. Figure 5.18 shows the percentile share for the group mobility models while Figure 5.19 shows the share for the entity mobility models.

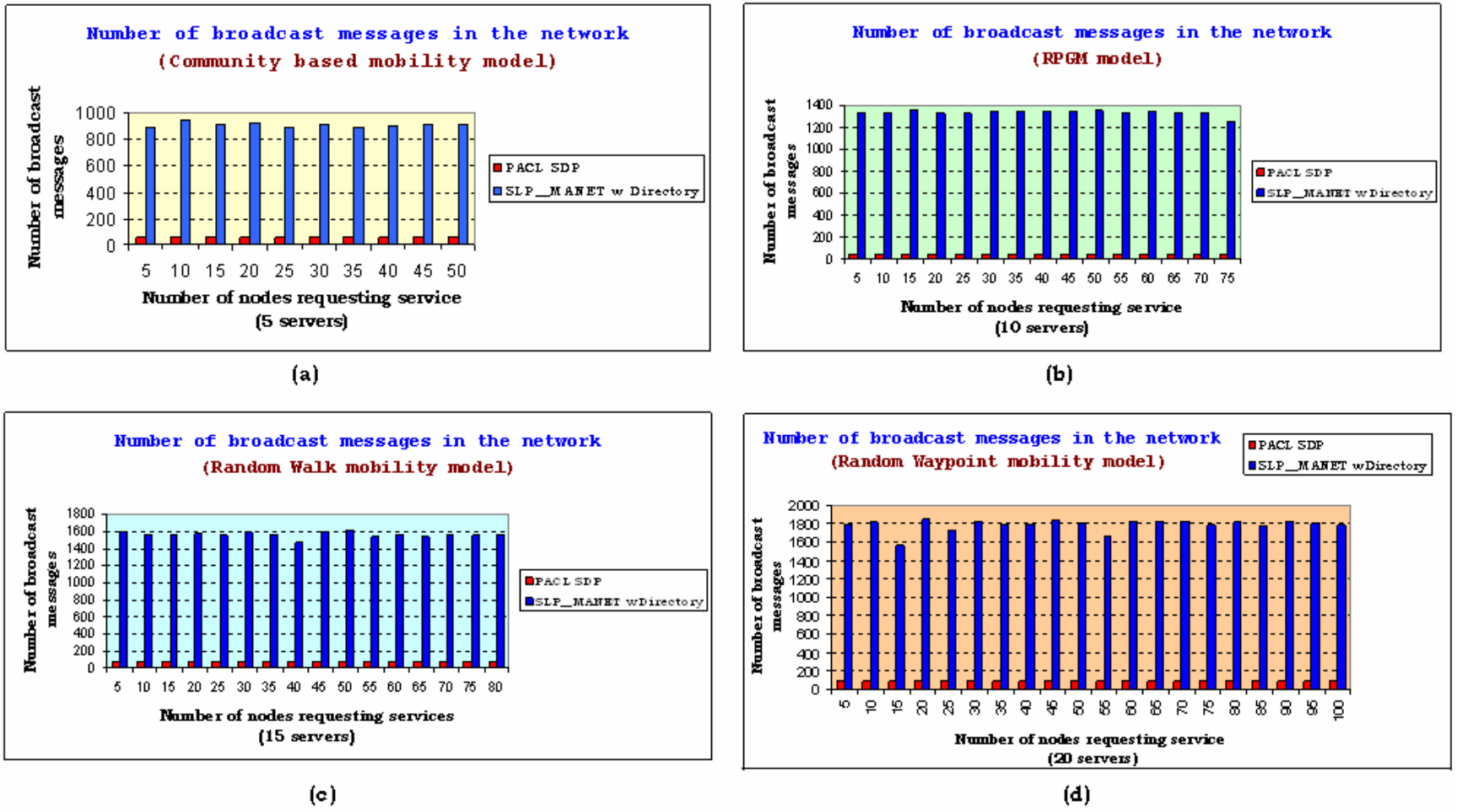

Fig. 5.16 Comparison of number of broadcast messages

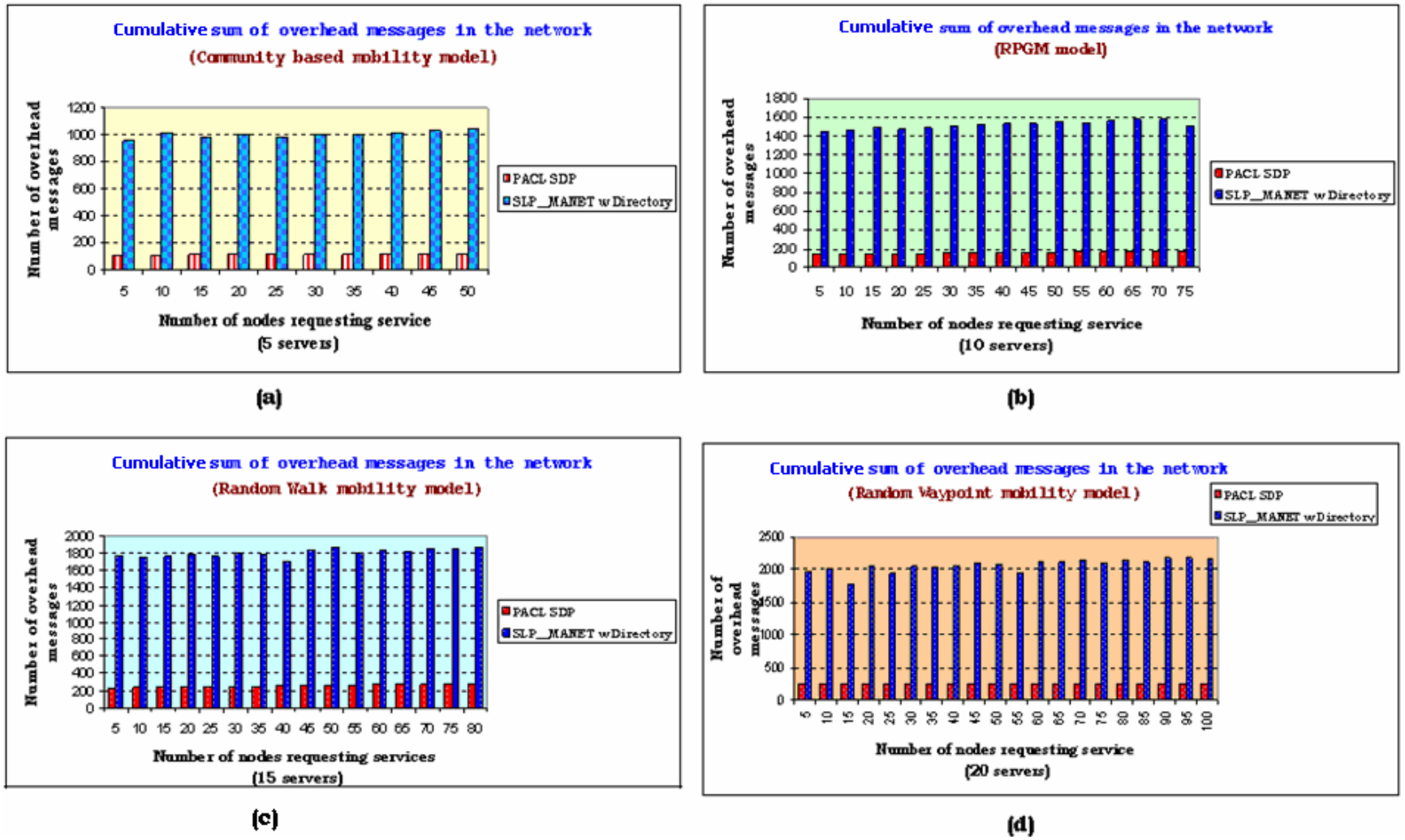

Fig. 5.17 Comparing cumulative of all types of overhead messages

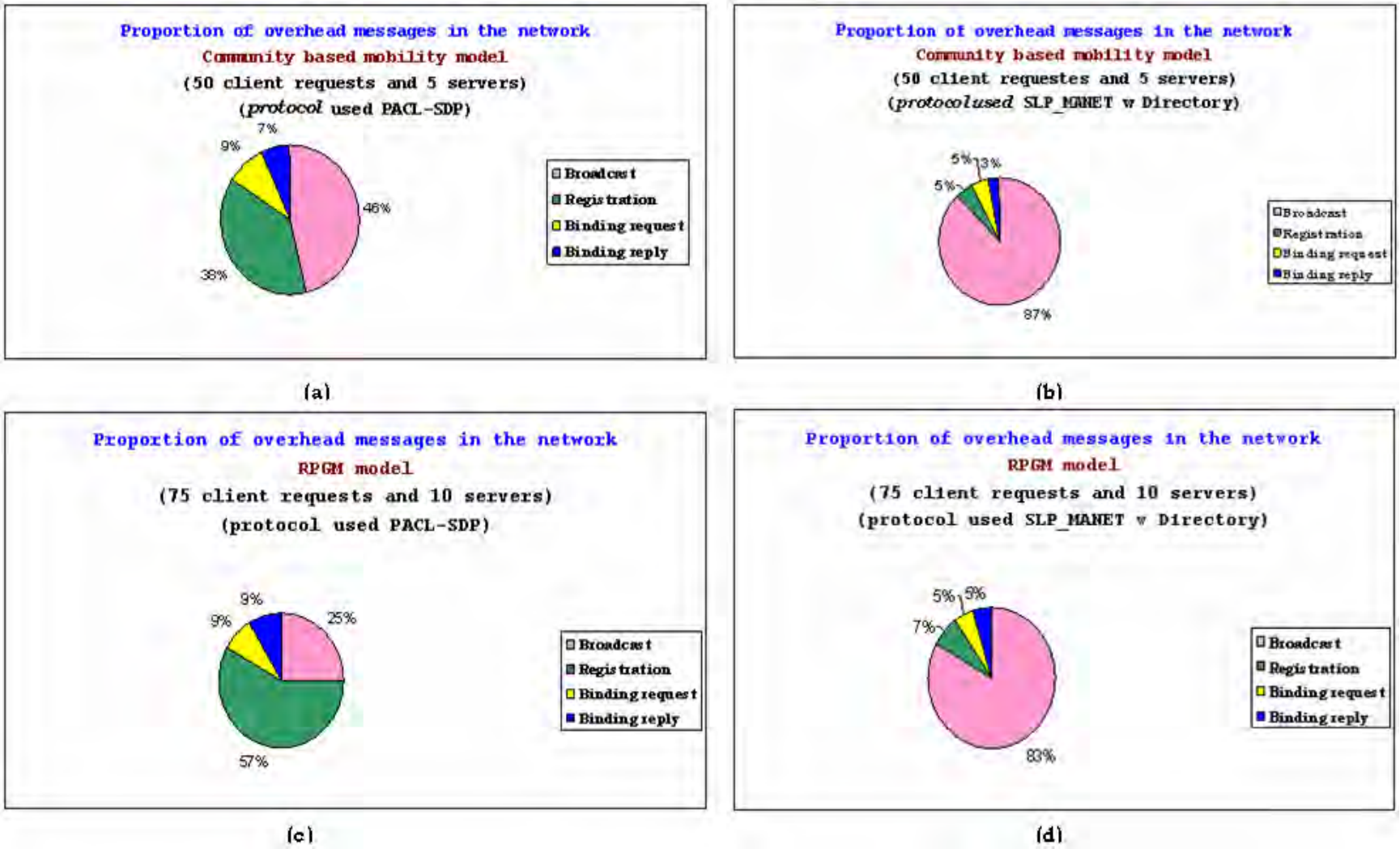

Fig. 5.18 Comparison of the proportion of overhead messages for the group mobility models

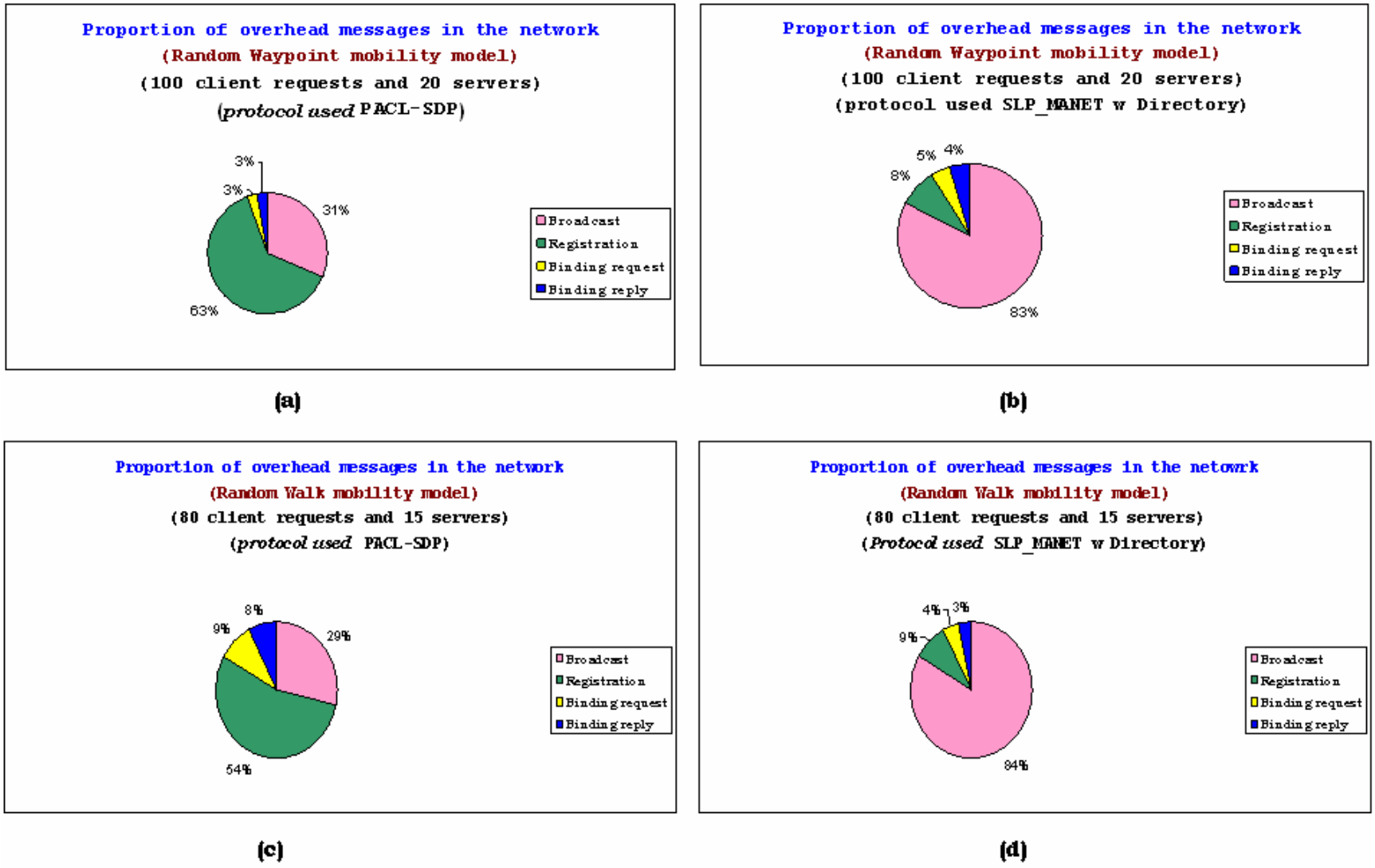

Fig. 5.19 Comparison of proportion of overhead messages for the entity mobility models

5.4.2 Comparative analysis of Network performance

i) Average energy consumption of client nodes

For this particular network performance metric the remaining energy of a client node after successful service discovery is used to indicate how efficiently energy is utilized by the underlying service discovery protocol. The average was computed as outlined in sub section 5.3.2. All member nodes in the cluster start with an initial energy amount of 1000.0 joules. As can be inferred from Figure 5.20 the network nodes can save considerable amount of energy for the cross-layered protocol. This is intuitive because for services that can be found at the vicinity of a client the client can discover the services with minimal energy consumption. However, using the traditional protocol a client has to discover a service through a directory node even if the service can be offered by its very neighbour.

ii) Energy consumption of a directory node

In both types of approaches the role of a directory node in the service discovery process is crucial. Yet, in the traditional approach the discovery process solely depends on the directory node. Every service discovery interaction passes through the registry node in terms of service binding request and the corresponding service binding reply. Hence, the registry node is forced to consume more energy in facilitating the process. On the other hand, for the cross-layered approach, the service of a lookup node is needed only if a client node can not find a service provider in its vicinity. Consequently, the directory node can save energy. The saving can be considerable when many client nodes within the cluster find themselves in the neighbourhood of servers. The comparison of energy consumption by a directory node under the two approaches is depicted in Figure 5.21. In the figure the available energy of the directory node as the function of the number of service requests is presented.

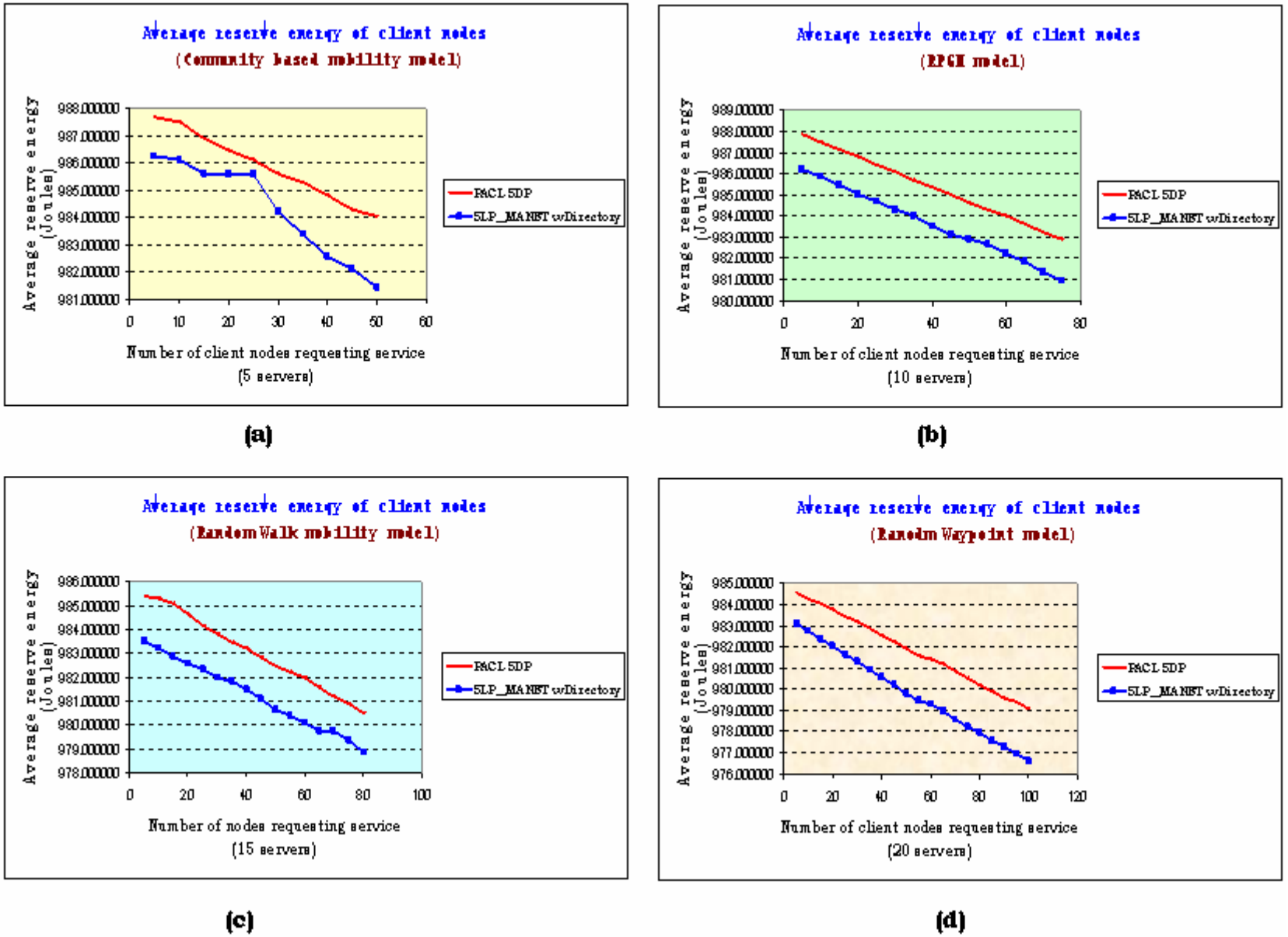

Fig. 5.20 Comparing the average energy consumption of client nodes

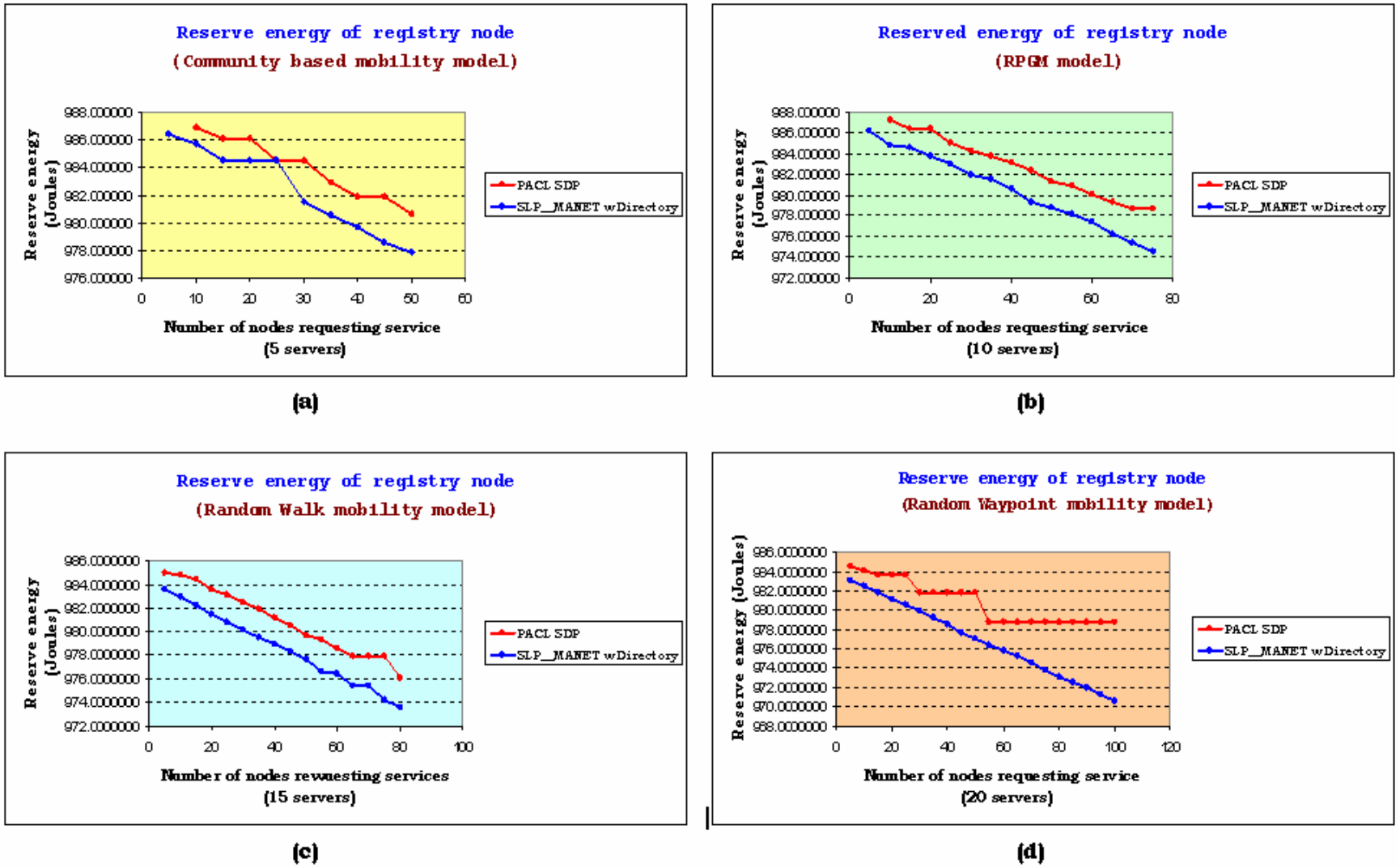

Fig. 5.21 Comparison of the energy consumption of a directory node

iii) Successful Service Discovery, SSD

In this research work the following metric was defined in measuring the success rate of a protocol to discover a service. It is assumed that there is a successful service discovery when a client node requesting for a service gets the first message from the requested server. And the metric that indicates this rate is the Successful Service Discovery, SSD, which is defined as the ratio of successfully discovered services to the number of requested services. Figure 5.22 shows that for the three mobility models considered; Community based mobility model, RPGM, and the Random Waypoint mobility model the cross-layered protocol has higher success rate than the traditional discovery protocol. For the random walk mobility model, traditional protocol has a higher success rate until a significant number of client nodes participate in the service discovery process. This can be explained as follows.

Servers may announce their capabilities to their neighbours. Neighbouring nodes then cache those advertisements. However, due to the erratic behaviour of the mobility model, either the clients themselves or the servers may disappear from the area unexpectedly. Now a client requiring a service tries to reach a server node whose address it has resolved from its neighbour's cache stored earlier. Yet, by that time the two nodes may be beyond the reachable radio range. This decreases the success of discovering the server sought. However, as several client nodes participate in the network, the success rate for the cross-layered protocol can go beyond that of the traditional protocol as illustrated in the same figure. This is not the case with the other types of mobility models as nodes in those models are characterized by features such as pause periods, and group motion, and hence are not unpredictable in their motion.

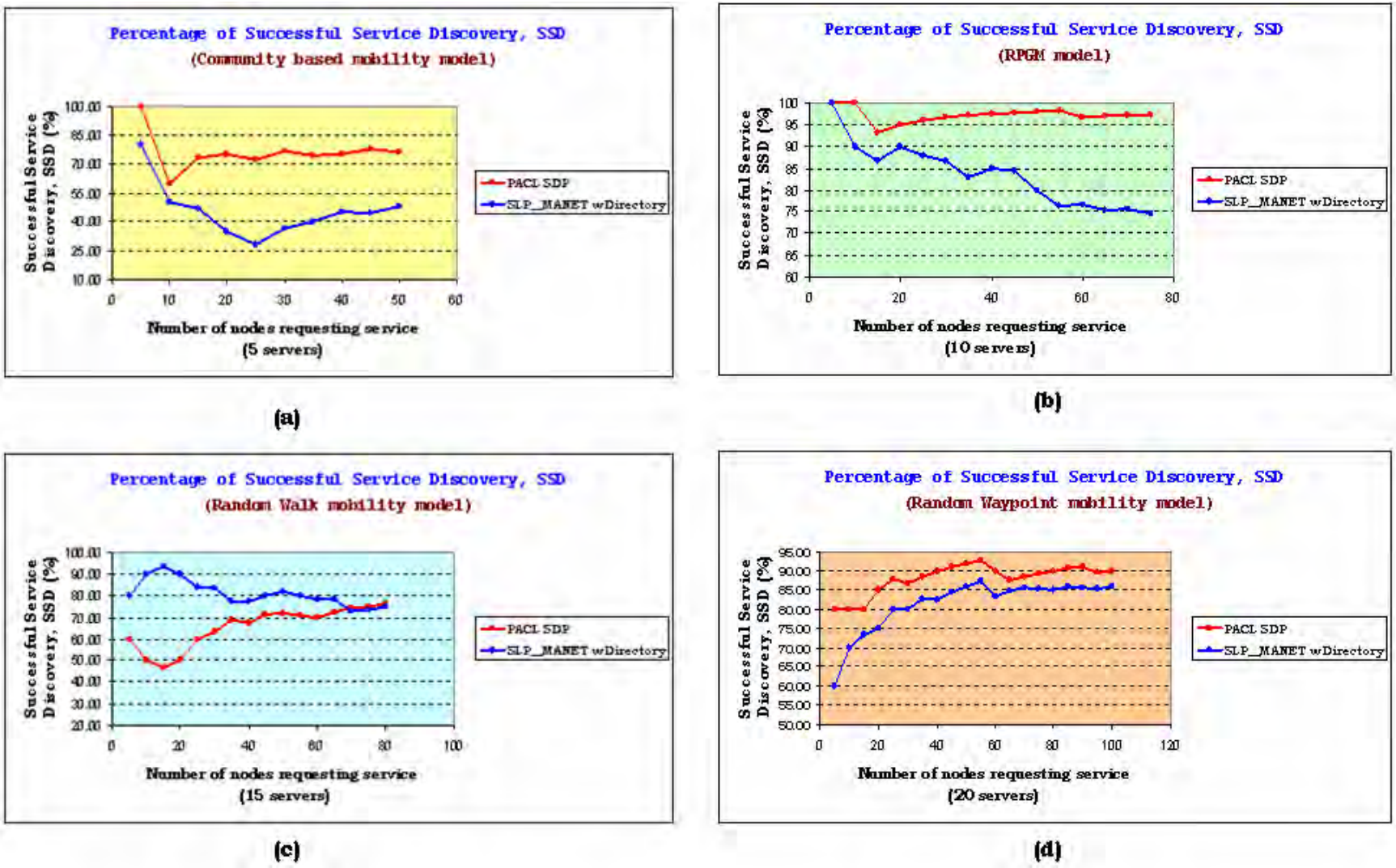

Fig. 5.22 Comparison of Successful Service Discovery rate, SSD

iv) Average response time

In this metric it is required to indicate the amount of time a client node has to wait before it receives the first response from the server node. To that end, it includes all latencies incurred during searching for the right service binding, route discovery to the server, and propagation delays during inter-node message transfers. As can be inferred from Figure 5.23 the cross-layered protocol exhibits less response time compared to the traditional discovery protocol as more clients are participating into the network. For the community based mobility model, when only few clients are active in the cluster, the probability of finding any near-by server that can provide the required service is less. Consequently, a client node may spend sometime searching its neighbours cache before it resorts to the cluster head. However, as more and more client nodes are actively participating in the network, several clients can invoke their services form servers around their neighbour. Consequently, with increasing number of client nodes, the performance of the cross-layered protocol gets better.

v) Packet delivery ratio, PDR

Packet delivery ratio can sometimes yield irrelevant information. Packets may arrive too late to be of any use to an application. These packets may be treated as packets that failed to arrive. However, the delivery ratio does not take this into consideration. To mitigate this problem in the results presented in this work a threshold of ten seconds was set for a packet to arrive before it is regarded as useless by an application. The reson behind this choice was because most of the packets have been observed to arrive in time period much less than ten seconds. Consequently, any packet arriving later than ten seconds can be treated as a failed packet. This was done in computation using the perl script shown in Appendix C**.** Based on this assumption, the PDR was computed for both protocols under the four mobility models and presented in Figure 5.24.

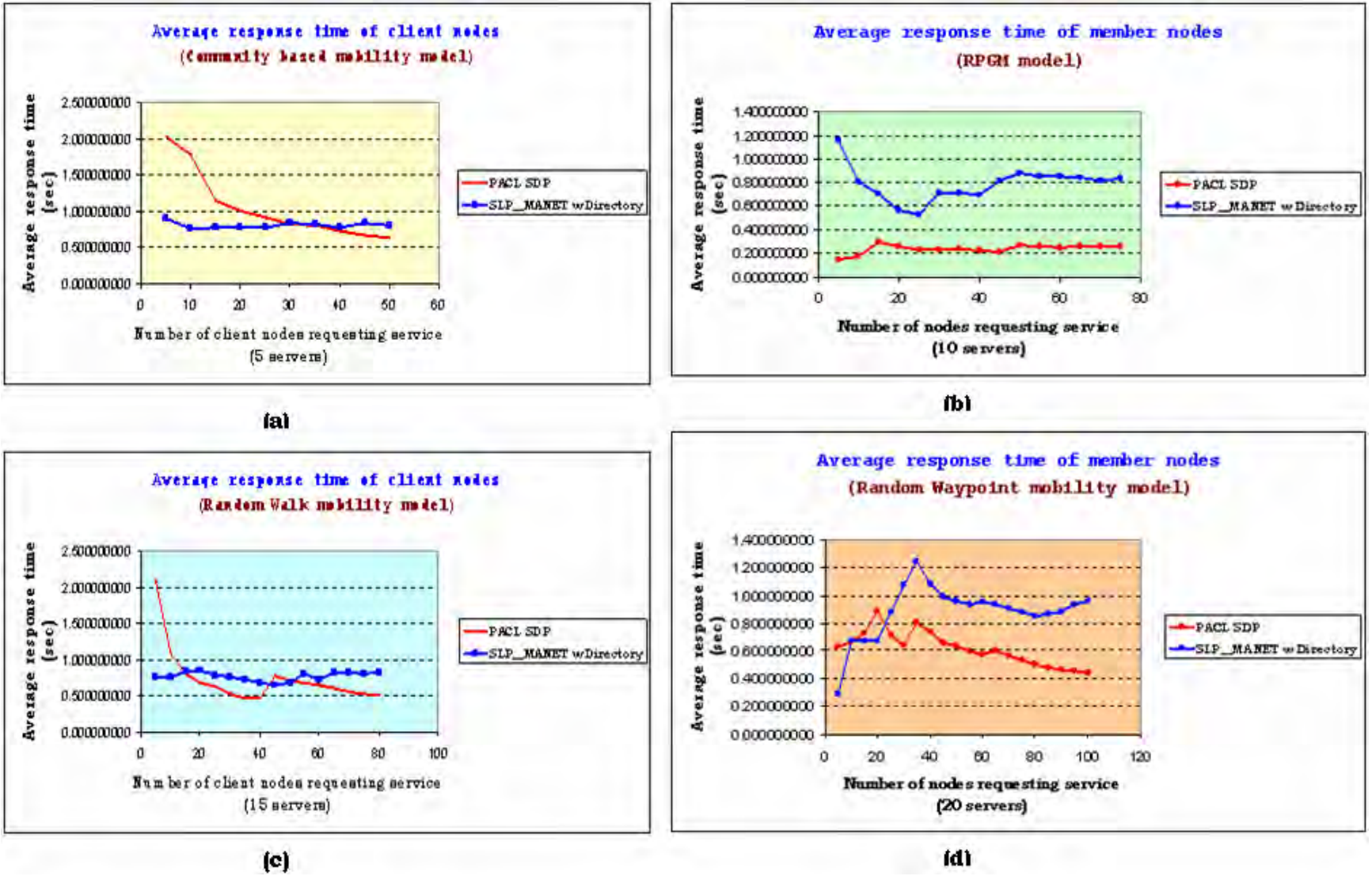

Fig. 5.23 Comparison of the average response time of client nodes for successful service discovery

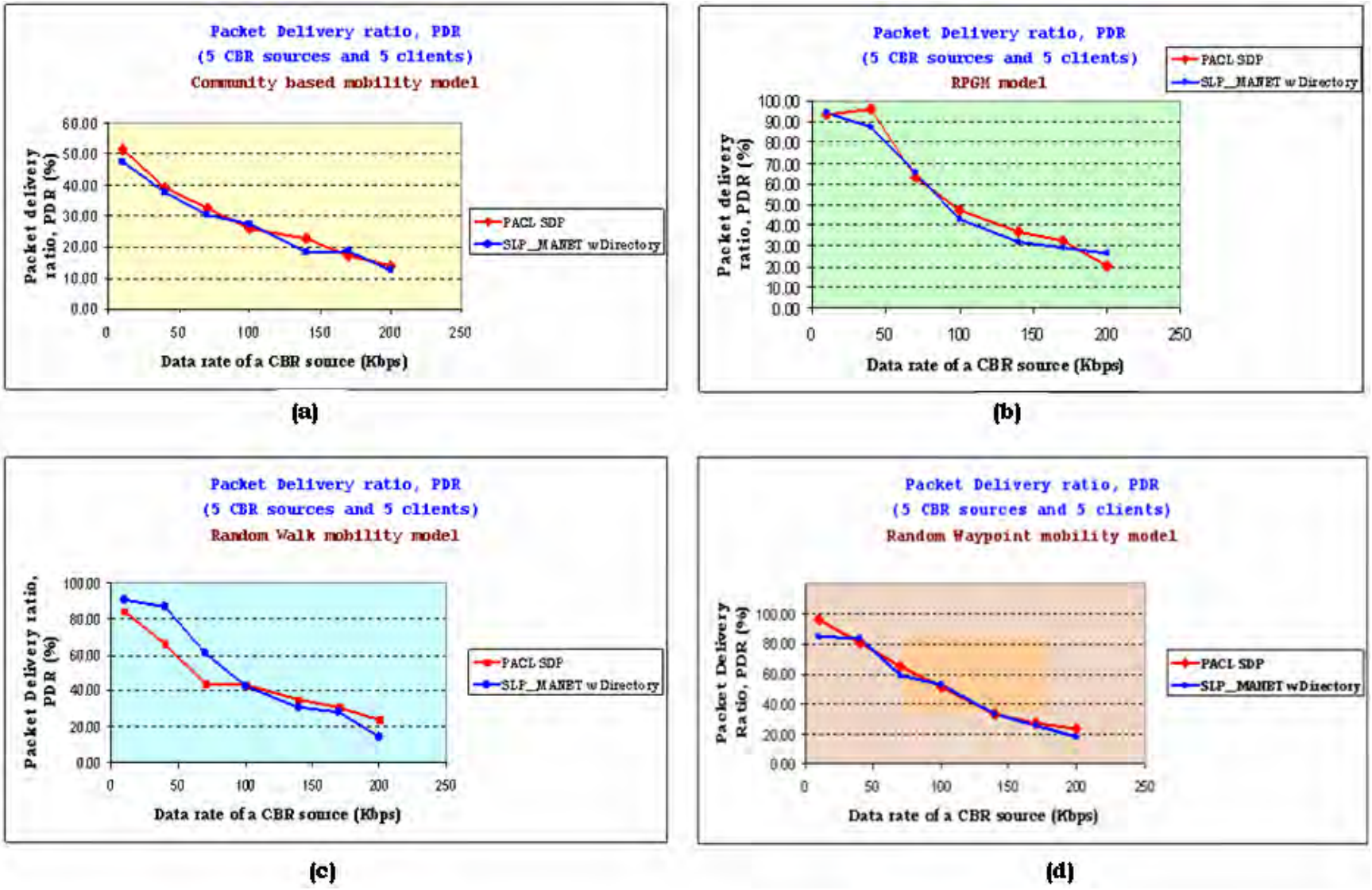

Fig. 5.24 Comparison of packet delivery ratio of 5 client nodes from 5 CBR sources

vi) Average throughput

Figure 5.25 shows the average throughput of 5 client nodes from 5 CBR sources for the different mobility scenarios. Any packet that arrives later than ten seconds is considered invalid. In computing the throughput a period of hundred seconds was used. This period was divided into ten divisions of each ten seconds interval. Then the evaluated throughput per node was the average of the computed throughput values over these intervals.

vii) Average end-to-end delay

The end-to-end delay incurred during the CBR packet transfer for the five CBR sources and their five clients is computed over different data rates. As as can be inferred from Figure 5.26 and Figure 5.22, that protocol which exhibits a lesser success ratio in discovering a service seems to have a lower end-to-end delay for the reception of CBR packets. This is expected because the network will not be as congested with service discovery messages as that protocol with higher successful discovery rate. Thus there is less delay incurred during the CBR transmission.

In addition, the protocol that attains a higher throughput and packet delivery ratio can incur a relatively higher end-to-end delay as the computation involves more packets. So this protocol that has featured more throughput and PDR, can exhibit higher average end-to-end delay.

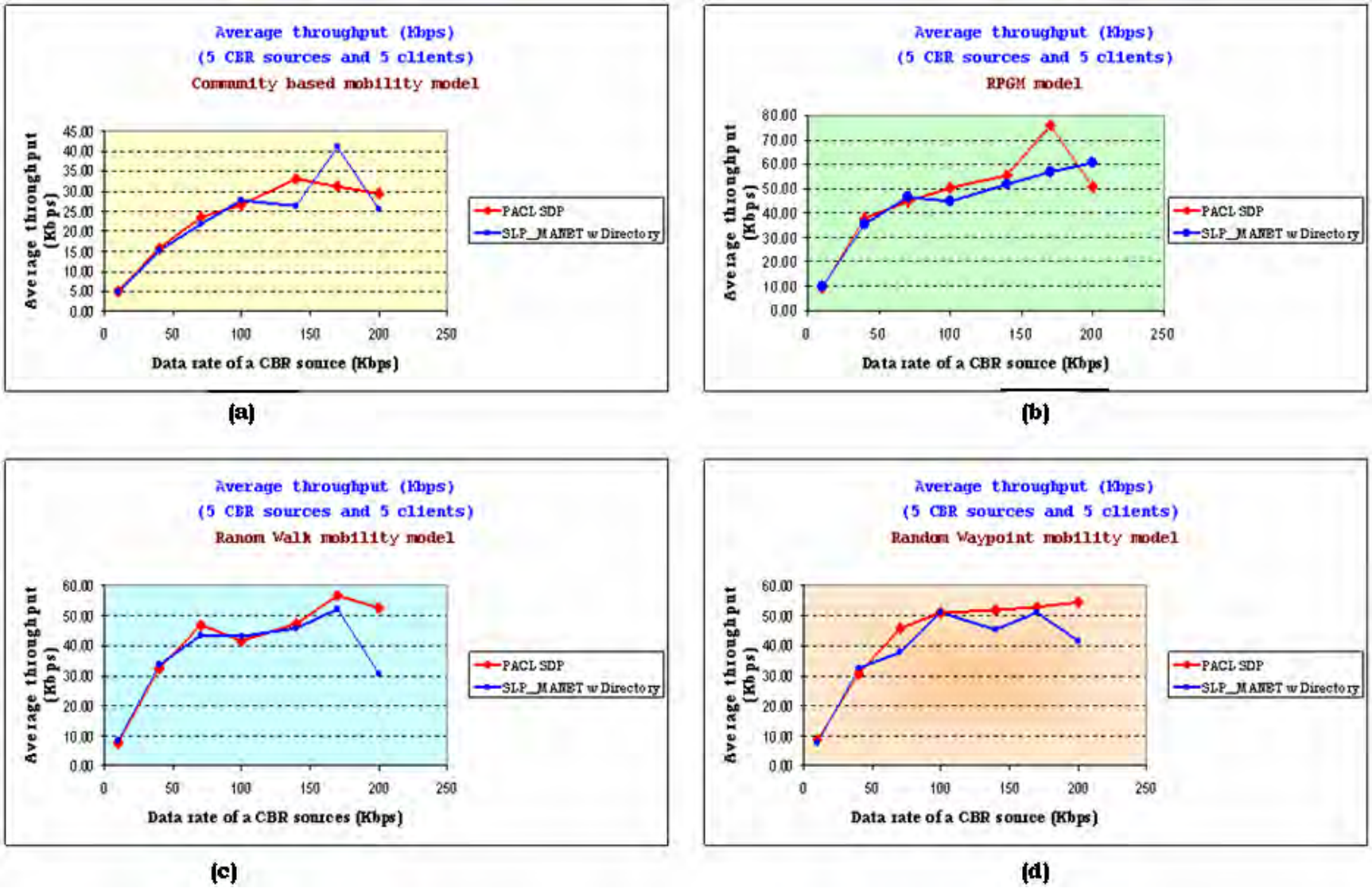

Fig. 5.25 Comparison of average throughput

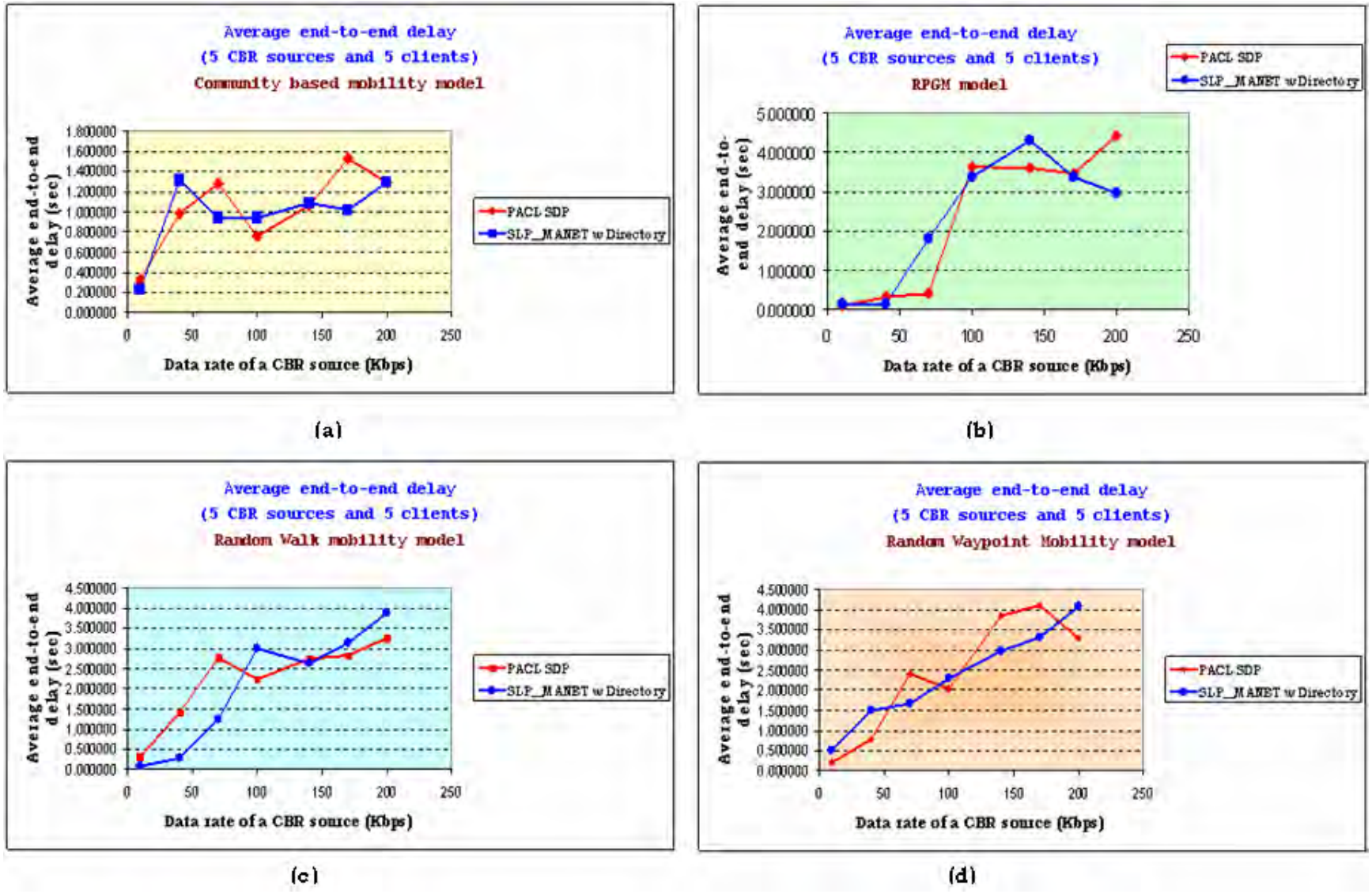

Fig. 5.26 Comparison of average end-to-end delay

5.5 Additional Simulation runs for Random Waypoint mobility model using different seed values

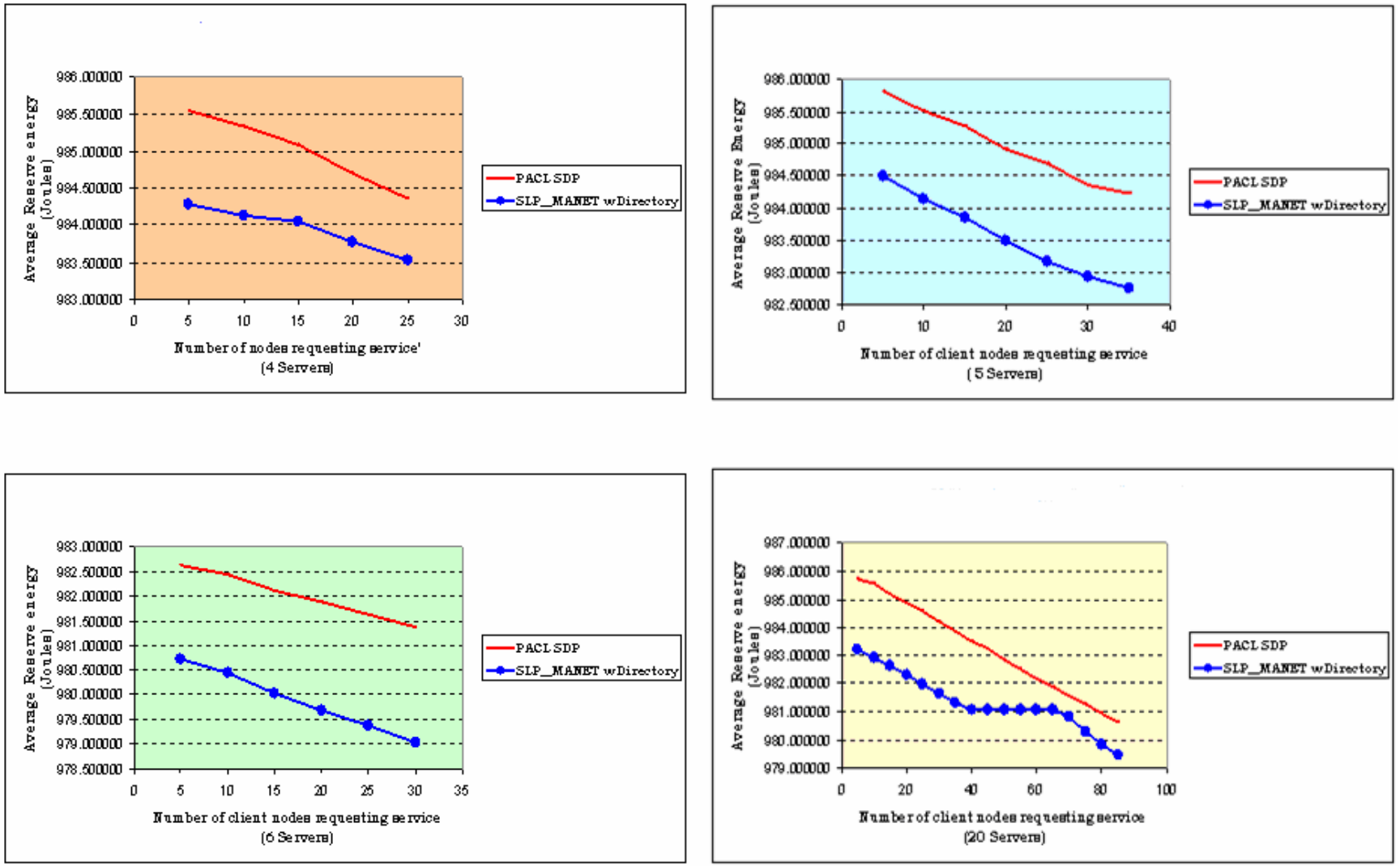

Fig. 5.27 Comparison of Average Reserve Energy of Client nodes for different cluster sizes

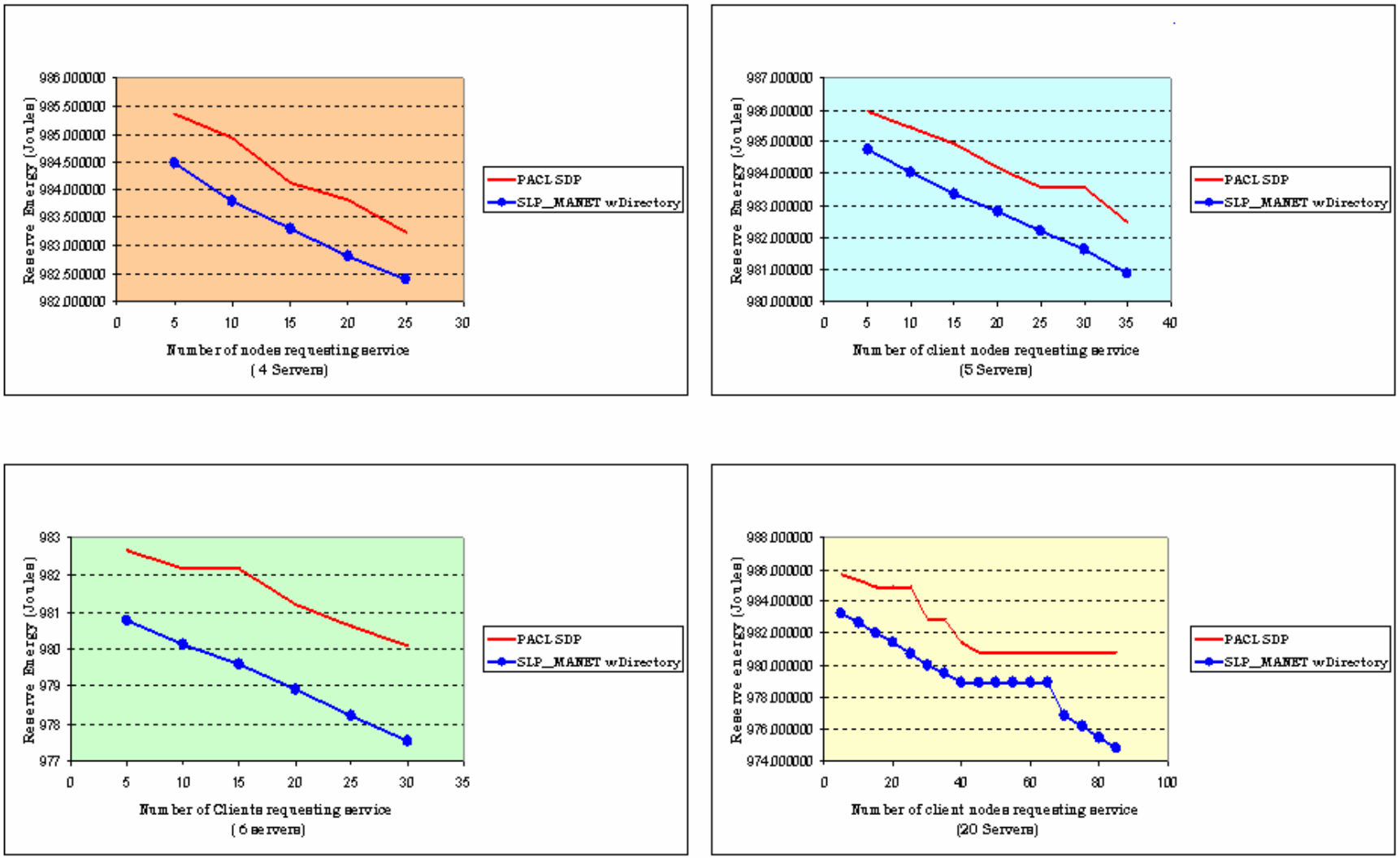

Fig. 5.28 Comparison of Reserve Energy of Directory node for different cluster sizes

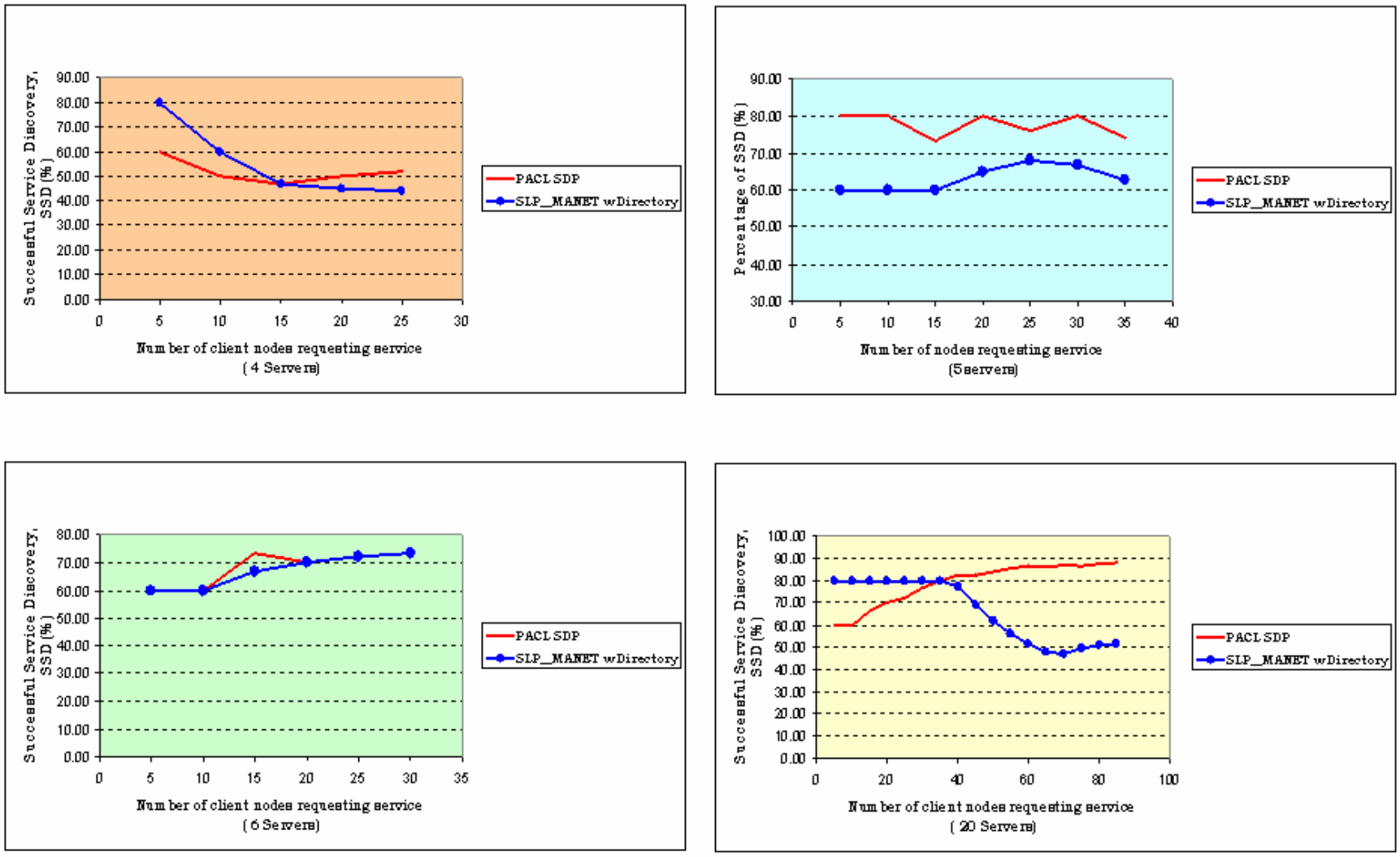

Fig. 5.29 Comparison of Successful Service Discovery, SSD for different cluster sizes

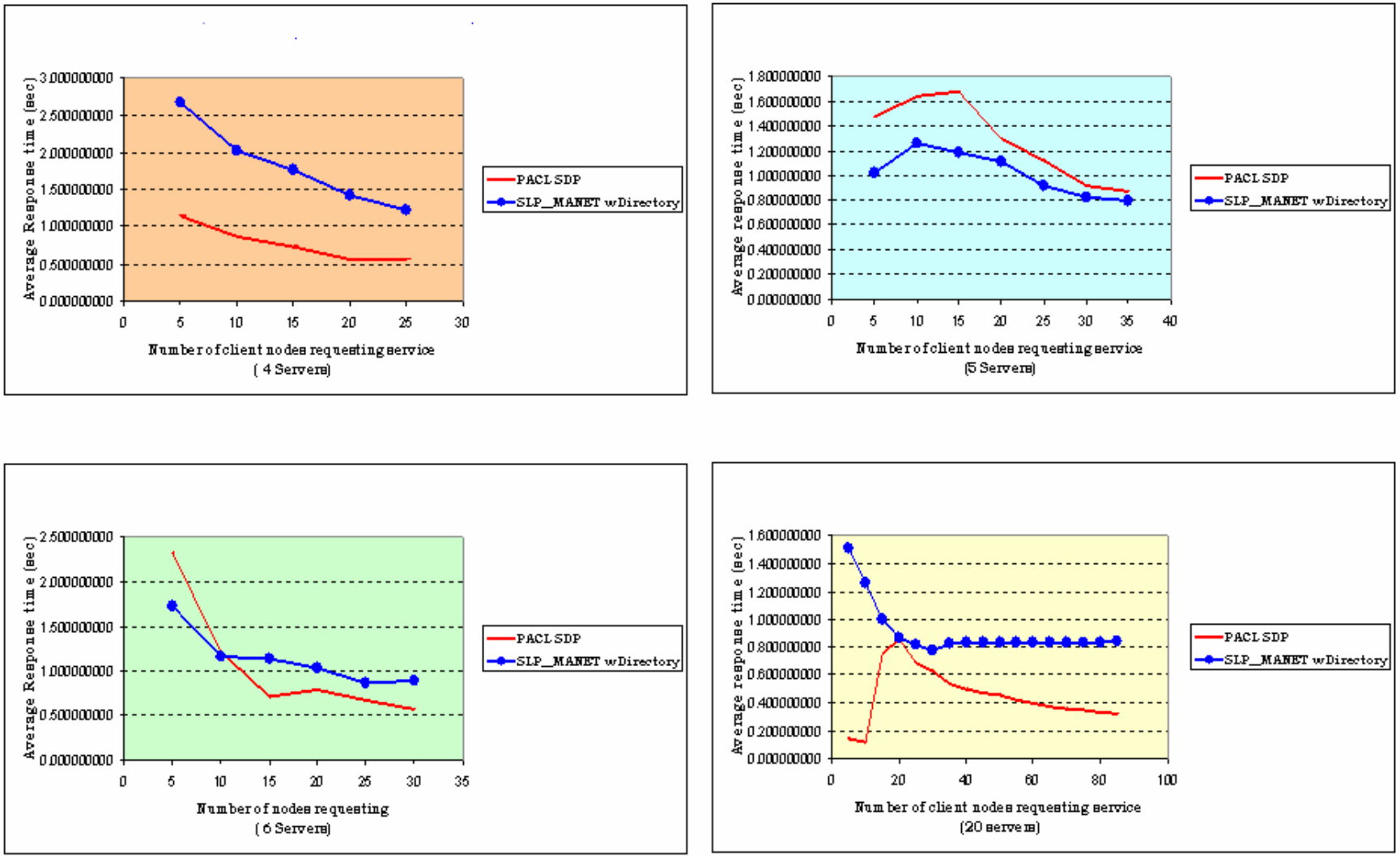

Fig. 5.30 Comparison of Average Response time of Client nodes for different cluster sizes

Chapter Six

Conclusion and Future work

6.1 Conclusion

The conclusions drawn from the results of this work are very convincing that the newly developed cross-layered service discovery architecture for mobile ad hoc networks, PACL-SDP, has a much better performance than the traditional modular approach under the various metrics considered. The major advantages of the new service discovery protocol over the traditional approaches are:

i) *Energy Saving*

It has been observed that the cross-layered protocol enjoys a considerable savings in energy consumption which stretches the life of a node to a longer period.

ii) *Increased service discovery rate*

The service discovery rate is very high as nodes are aware of services available in their neighbourhood.

iii) *Lower discovery time*

Nodes can discover services in a lesser time period than possible in the traditional approach.

iv) *Scalability*

The new architecture is highly scalable to large network sizes. This is the consequence of the fact that the protocol does not rely on any broadcast messages during service discovery or service acquisition.

v) *Improved packet delivery ratio*

The new protocol has a higher packet delivery ratio than the traditional protocol.

vi) *Increased throughput*

The new architecture brings an increase in the throughput of data transfers, as the protocol does not involve much broadcast messages.

vii) *Reduced overhead message*

The new solution incurs far less congestion due to service discovery overhead messages.

Table 6.1 below summarizes the average performance of PACL-SDP with reference to the directory based SLP_MANET service discovery protocol for the scenarios considered.

Table 6.1 Performance of the cross-layered protocol over the traditional protocol

Performance metric	*Mobility Model*			
	Community	*RPGM*	*Random Waypoint*	*Random Walk*
SSD	**67.47% More**	**16.38% More**	**8.14% More**	**19.12% Less**
Response time	**29.18% More**	**69.67% Less**	**30.25% Less**	**4.057% Less**
Energy saving	**10.39% More**	**11.4% More**	**13.73% More**	**10.24% More**
Packet delivery ratio	**5.52% More**	**3.49% More**	**5.76% More**	**7.87% Less**
Throughput	**3.8 % More**	**4.62% More**	**8.91% More**	**2.64% More**
End-to-end delay	**6.1% More**	**16.86% More**	**5.274% More**	**4.674% More**
Registration overhead	**8.16% Less**	**8.93% Less**	**10.91% Less**	**9% Less**
Broadcast overhead	**93.97% Less**	**96.70% Less**	**95.46% Less**	**94.94% Less**
Binding request/reply overhead	**75.64% Less**	**79.31% Less**	**92.82% Less**	**67.59% Less**

Note that metrics SSD, Energy saving, packet delivery ratio, and throughput are HB (Higher is better) metrics, while response time and end-to-end delay and the overhead types are LB (lower is better) metrics.

The overhead messages considered are for the case where all the memebers of the cluster are actively participating in service discovery processes.

From the table it can be observed that for the community based mobility model the response time and end-to-end delay for the new cross-layered architecture are relatively larger than that observed for the traditional protocol. This is the consequence of the fact that the new protocol has a much higher service discovery rate and hence much more nodes have active interaction in resource discovery and acquisition than the traditional case. Hence, this could incur some delay.

Also it can be seen that for Random Walk mobility model the cross-layered architecture produced less discovery rate. As explained in the previous chapter this can be the result of the unpredictable behaviour of nodes in this mobility model. Server nodes announce their capabilities to their neighbours and either the client nodes or the server itself vanish from the area. Consequently the client nodes can not reach the servers when they need them. A possible mitigation step that can be used is to allow a client node to resort to the directory node if no response comes from the supposedly neighbouring server in a given time period. This can raise the discovery rate though it could increase the response time.

6.2 Future work

The study carried out in this thesis work has focused on service discovery for mobile nodes forming a cluster. Hence, the discovery process considered is what is referred to as intra-cluster service discovery process. As such, it is possible to extend the work to incorporate inter-cluster service discovery processes. This helps nodes to discover services outside their domain.

The work for the extension of the algorithm may be implemented without much effort as the considered routing layer protocol, Ad hoc On-Demand Distance Vector, AODV, possesses a multicast extension capability, MAODV [16]. Using this feature, cluster heads within the network can form an overly network among their peers. Then, any request for service that can not be resolved by a cluster head can be multicasted over the overlay network. This way all available resources within the network can be reachable by all client nodes.

Another possible extension is to incorporate a secondary node within a cluster. This helps in maximizing service availability by providing a fallback directory node for members of a cluster in case the primary node fails or leaves the network.

Referrences:

[1] Adnan Noor Mian, Roberto Beraldi, Roberto Baldoni, "Survey of Service Discovery Protocols in Mobile Ad Hoc Networks," Dipartimento di Informatica e Sistemistica "Antonio Ruberti" Università degli Studi di Roma "La Sapienza" Via Salaria, 113 – 00198 Rome, Italy Technical Report, 2006

[2] Gregor Schiele, Christian Becker, "SANDMAN: an Energy-Efficient Middleware for Pervasive Computing", Information Systems II Universität Mannheim Schloss, 68131 Mannheim, Germany 2007.

[3] Reinhard Diestel, *Graph Theory,* New York: Springer-Verlag Heidelberg, [E-book], 2005.

[4] A. Velmurugan and R. Rajaram, "Adaptive Hybrid Mobile Agent Protocol for Wireless Multihop Internet Access," *Journal of Computer Science,* vol. 2, no.9, 676-682, 2006 ISSN 1549-3636 © 2006 Science Publications.

[5] Christopher N. Ververidis and George C. Polyzos, Members, IEEE, "Service Discovery for Mobile Ad Hoc Networks: A Survey of Issues and Techniques," This research has been supported in part by the project Autonomic Network Architecture (ANA, #IST-27489), which is funded by the IST FET Program of the European Commission, 2008

[6] Mohamed M. Abou El Saoud, M.S.thesis, "MANET Reference Configurations and Evaluation of Service location protocol for MANETs," Ottawa-Carleton, Institute for Electrical and Computer Engineering, Carleton University Ottawa, Ontario Canada, April 2005.

[7] Subir Kumar Sarkar, T G Basavaraju, C. Puttamadappa, *Ad hoc Mobile Wireless Networks,* New York : Auerbach Publications, 2008.

[8] Kayhan Erciyes, Orhan Dagdeviren, Deniz Cokuslu, "Modelling and Simulationof Wireless Sensor and Mobile Ad Hoc Networks," *Proceedings of the International Conference on Modeling and Simulation,* 28 – 30 August 2006, Konya, TURKEY.

[9] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research Wireless Communication & Mobile Computing (WCMC)," *Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications,* vol. 2, no. 5, pp 483-502, 2002.Copyright (C) 2004 Toilers Research Group -- Colorado School of Mines.

[10] Mirco Musolesi, Cecilia Mascolo, "Designing Mobility Models based on Social Networks Theory," Department of Computer Science, University College London, United Kingdom. *Mobile Computing and Communications Review*, vol.1, no.2, July 2007.

Generator for NS 2 simulator Based on the Community Based Mobility Model Version 0.3 December 2006, Copyright (C) Mirco Musolesi University College London 2006-2008.

[11] C K Toh, Chai K Toh, *AD HOC MOBILE WIRELESS NETWORKS, Protocols and Systems,"* New Jersey : Prentice Hall PTR, 2001.

[12] Carlos de Morais Cordeiro, Dharma Prakash Agrawal, *Ad hoc & Sensor networks,* Singapore : World Scientific Publishing Co. Pte. Ltd, 2006.

[13] Dante Arias-Torres, José Antonio García-Macias, "Service Discovery in Mobile Ad-hoc Networks by Extending the AODV Protocol," Departamento de Ciencias de la Computación Centro de Investigación Científica y Educación Superior de Ensenada, CICESE, Km. 107 Carretera Tijuana – Ensenada, CP 22860, Ensenada, B.C. México, 2005.

[14] Christopher N. Ververidis and George C. Polyzos, Member, IEEE, "AVERT: Adaptive SerVicE and Route Discovery ProTocol for MANETs," Computer Science Department, Athens University of Economics and Business, Athens, Greece, 2008.

[15] NS 2 Newtork simulator. http://www.isi.edu/nsnam/ns/

[16] Elizabeth M. Royer, University of California, Santa Barbara, Charles E. Perkins Nokia Research Center, "Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing draft-ietf-manet-maodv-00.txt," Mobile Ad Hoc Networking Working Group. INTERNET DRAFT.15 July, 2000. Work in progress.

[17] Sudarshan Vasudevan, Jim Kurose, Don Towsley, "Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks," Department of Computer Science University of Massachusetts, Amherst, 2003.

[18] Muhammad Mizanur Rahman, M. Abdullah-Al-Wadud, Oksam Chae, "Performance analysis of Leader Election Algorithms in Mobile Ad hoc Networks," *IJCSNS International Journal of Computer Science and Network Security*, vol..8 no.2, February, 2008.

[19] Vaskar Raychoudhury, Jiannong Cao, Weigang Wu, "Top K-leader Election in Wireless Ad Hoc Networks," Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong. ©2008 IEEE.

[20] Alex Varshavsky, Bradley Reid, Eyal de Lara, "A Cross-Layer Approach to Service Discovery and Selection in MANETs," Department of Computer Science University of Toronto. 2005.

[21] E.W. Dijkstra and C.S. Scholten, "Termination detection for diffusing computations," *In Information Processing Letters*, vol. 11, no. 1, pp. 1-4, copyright © 1980.

[22] Steffano Basagni, Marco Conti, Silvia Giordano Ivan Stojmenovic, *Mobile Ad hoc Networking,* New Jersey : A John Wiley & Sons, inc. publication, 2004.

[23] Andrei Broder, IBM Research Division, Hawthorne, NY 10532, Michael Mitzenmacher, Harvard University, Division of Engineering and Applied Sciences, Cambridge, MA 02138. "Network Applications of Bloom Filters: A Survey," May 10, 2004.

[24] Salutation Consurtium, *Salutation Architecture Specification (Part- 1) ver 2.0c.* © Copyright The Salutation Consortium Inc. June 1, 1999.

[25] http://developer.apple.com/networking/bonjour/index.html

[26] Jan Newmarch, *Foundations of Jini 2 Programming,* New York : Apress © 2006.

[27] Alan Presser et al, *UPnP Device Architecture 1.1,* Document Revision Date 15 October 2008, Contributing Members of the UPnP Forum. © 2008.

[28] Ivan Stojmenovic, Ed, *Handbook of Wireless Networks and Mobile Computing,* New York : John Wiley & Sons, Inc.Copyright 2002.

[29] E. Guttman ,C. Perkins,. J.Veizades, M.Day Vinca Coprporation, "Service Location Prtotocol, version 2," Network Working Group. Request for Comments: 2608. Category: Standards Track. June 1999.

[30] A. M. Ortiz, T. Olivares, F. Royo, P. Díaz, and L. Orozco-Barbosa, "A cross-layer integration prototype for wireless sensor networks," Albacete Research Institute of Informatics University of Castilla-La Mancha (UCLM), Spain 2009.

[31] R. Koodli and C. E. Perkins, "Service discovery in on-demand ad hoc networks," IETF Internet Draft, draft-koodli-manet-servicediscovery- 00.txt, October 2002. Work in progress.

[32] Joakim Flathagen, Norwegian Defence Research Establishment, Kjeller, Norway. Knut Øvsthus, Bergen University College, Bergen, Norway, "Service Discovery using OLSR and Bloom Filters," *In Proceedings of the 4th OLSR Interop & Workshop*, October 14-16, 2008.

[33] Charles E. Perkins, Nokia Research Center, Elizabeth M. Belding-Royer, University of California, Santa Barbara, S.Das. Univesity of Cincinnaty, "Ad hoc On-Demand Distance Vector (AODV) Routing," Network Working Group. RFC 3561.Category: Experimental. July 2003.

[34] Olga Ratsimor, Dipanjan Chakraborty, Anupam Joshi, Timothy Finin, "Allia: Alliancebased Service Discovery for AdHoc Environments," Department of computer science and Electrical engineering. University of Maryland, Atlanta, Georgia, USA. Copyright © ACM. September 2002.

[35] Sumi Helal, Nitin Desai, Varun Verma and Choonhwa Lee, "Konark – A Service Discovery and Delivery Protocol for Ad-Hoc Networks," Computer and Information Science and Engineering Department University of Florida, Gainesville, FL-32611, USA, 2003.

[36] Feng Zhu, Matt Mutka, Department of Computer Science and Engineering,Michigan State University. Lionel Ni1, Department of Computer Science,Hong Kong University of Science and Technology, "Splendor: A Secure, Private, and Location-aware Service Discovery Protocol Supporting Mobile Services," *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications* (PerCom 2003).

[37] Michael Klein Birgitta Konig-Ries Philipp Obreiter, "Service Rings – A Semantic Overlay for Service Discovery in Ad hoc Networks," Institute for Program Structures and Data Organisation Universit¨at Karlsruhe D-76131 Karlsruhe, Germany 2003.

[38] Rekha Patil, Gulbarga. Dr A.Damodaram, "Cost Based Power Aware Cross Layer Routing Protocol For Manet," *IJCSNS International Journal of Computer Science and Network Security*, vol.8 no.12, December 2008.

[39] Didem Gozupek, Nirwan Ansari, "A Cross-Layer Architecture for End-to-End QoS Provisioning in Wireless Ad Hoc Networks," Department of Electrical and Computer Engineering, New Jersey Institute of Technology, U.S.A, Symeon Papavassiliou, School of Electrical and Computer Engineering, National Technical University of Athens. Greece, 2006.

[40] Shafiullah Khan, Kohat University of Science and Technology (KUST), Pakistan, Zia Ud Din, Gomal University, Pakistan, Kok Keong Loo, School of Engineering and Design, Brunel University, UK, "Cross Layer Design for Routing and Security in Multi-hop Wireless Networks," *Journal of Information Assurance and Security* 4 (170-173), 2009.

[41] C. Hager, D. J. Shyy, and J. Ma, "Cooperative Cross-Layer Design for Wireless Networks," *Journal of communications*, vol. 3, no. 4, SEPTEMBER 2008.

[42] José Villalón, Pedro Cuenca, Luis Orozco-Barbosa, Yongho Seok, and Thierry Turletti. "Cross-Layer Architecture for Adaptive Video Multicast Streaming Over Multirate Wireless LANs," *IEEE Journal on selected areas in communications* vol. 25, no. 4, MAY 2007.

[43] Emmanuel Baccelli, Philippe Jacquet, "Flooding Techniques in Mobile Ad Hoc Networks," Unité de recherche INRIA Rocquencourt. Rapport de recherche n° 5002 _ November 2003.

[44] Donggeon Noh and Heonshik Shin, "SPIZ: An Effective Service Discovery Protocol for Mobile Ad Hoc Networks," *Journal on Wireless Communications and Networking,* Article ID 25167, 13 pages Volume 2007.

[45] Nor Surayati Mohamad Usop, Azizol Abdullah, Ahmad Faisal Amri Abidin. "Performance Evaluation of AODV, DSDV & DSR Routing Protocol in Grid Environment," *IJCSNS International Journal of Computer Science and Network Security*, vol.9 no.7, July 2009.

[46] Anuj K. Gupta, Member, IACSIT, Dr. Harsh Sadawarti, Dr. Anil K. Verma, "Performance analysis of AODV, DSR & TORA Routing Protocols," *IACSIT International Journal of Engineering and Technology*, vol.2, no.2, April 2010.

Appendix A

TCL scripts for the Simulation

A.1 TCL Script defining network setup

 # Create the Transport layer protocols

```
for {set i 0} {$i < $val(nn) } {incr i} {
   set ls_agent_($i) [new Agent/UDP/LS_Agent]

   }
```

 # Attaching Agents to nodes

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

      $ns_ attach-agent $node_($i) $ls_agent_($i)
   }
```

 # Create the Application layer protocols

```
for {set i 0} {$i < $val(nn) } {incr i} {
   set ls_app_($i) [new Application/LS_App]

   }
```

 # Set firing time for leader election

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

      $ls_agent_($i) set-time 100
   }
```

 # Attach application layer to transport layer

```
for {set i 0} {$i < $val(nn) } {incr i} {
    $ls_app_($i) attach-agent $ls_agent_($i)
}
```

 # Connect transport layer agents

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

     for {set j [expr $i + 1] } {$j < $val(nn) } {incr j} {

      $ns_ connect $ls_agent_($i) $ls_agent_($j)
   }
}
```

```
# Sample code for seting up services at application layer of nodes

$ns_ at 205.1 "$ls_app_(50) set-service PRINT"
$ns_ at 206.3 "$ls_app_(67) set-service GATE_WAY"
$ns_ at 207.5 "$ls_app_(16) set-service SENSOR"
$ns_ at 208.7 "$ls_app_(98) set-service FTP"
$ns_ at 209.9 "$ls_app_(69) set-service STORAGE"

# Sample code when nodes request for a service

$ns_ at 260.1 "$ls_app_(31) request-service PRINT"
$ns_ at 260.3 "$ls_app_(40) request-service GATE_WAY"
$ns_ at 260.5 "$ls_app_(11) request-service SENSOR"
$ns_ at 260.7 "$ls_app_(46) request-service STORAGE"
$ns_ at 260.9 "$ls_app_(20) request-service FTP"

# Sample code for setting up CBR traffic for selected clients

for {set i 0} {$i < 5 } {incr i} {
   set cbr($i) [new Application/Traffic/CBR]

   $cbr($i) set type_ CBR
   $cbr($i) set packet_size_ 1024
   $cbr($i) set rate_ 170kb
   $cbr($i) set random_ false
 }

#$ns_ at 208.0 "$ls_agent_(59) set my_client 1"
#$ns_ at 208.0 "$ls_agent_(82) set my_client 103"
#$ns_ at 208.0 "$ls_agent_(110) set my_client 37"
#$ns_ at 208.0 "$ls_agent_(107) set my_client 25"
#$ns_ at 208.0 "$ls_agent_(115) set my_client 130"

#$ns_ at 208.0 "$cbr(0) attach-agent $ls_agent_(59)"
#$ns_ at 208.0 "$cbr(1) attach-agent $ls_agent_(82)"
#$ns_ at 208.0 "$cbr(2) attach-agent $ls_agent_(110)"
#$ns_ at 208.0 "$cbr(3) attach-agent $ls_agent_(107)"
#$ns_ at 208.0 "$cbr(4) attach-agent $ls_agent_(115)"

$ns_ at 209.1 "$cbr(0) start"
$ns_ at 209.3 "$cbr(1) start"
$ns_ at 209.3 "$cbr(1) start"
$ns_ at 209.5 "$cbr(3) start"
$ns_ at 209.5 "$cbr(4) start"

$ns_ at 309.1 "$cbr(0) stop"
$ns_ at 309.3 "$cbr(1) stop"
$ns_ at 309.5 "$cbr(2) stop"
$ns_ at 309.3 "$cbr(3) stop"
$ns_ at 309.5 "$cbr(4) stop"

# Start running the applications

for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 2.0 "$ls_app_($i) start"
}
```

A.2 Tcl Script defining Scenarios and network configurations

```
set val(chan)          Channel/WirelessChannel        ;# channel type
set val(prop)          Propagation/TwoRayGround       ;# radio-propagation model
set val(ant)           Antenna/OmniAntenna            ;# Antenna type
set val(ll)            LL                             ;# Link layer type
set val(ifq)           Queue/DropTail/PriQueue        ;# Interface queue type
set val(ifqlen)        50                             ;# max packet in ifq
set val(netif)         Phy/WirelessPhy                ;# network interface type
set val(mac)           Mac/802_11                     ;# MAC type
set val(rp)            AODV                           ;# ad-hoc routing protocol
set val(nn)            100                            ;# number of mobilenodes
set val(energyMdl)     EnergyModel
set val(TxPwr)         0.281838
set val(initenergy)    1000
set val(x)             500                            ;# x dimension of topography
set val(y)             500                            ;# y dimension of topography
set val(stop)          500                            ;# Simulation end time)
;#set val(seed)        0.0                            ;# seed instantiating . Does not
work!
set val(sc)            rpgm_100                       ;# Scenario file
set val(cp)            Directory_energy               ;# The traffic pattern file


;#$defaultRNG seed 100 ;# If one wants to alter the seed value make a change here

set ns_ [new Simulator]
set traceFile [open normal.tr w]
set namtraceFile [open normalNAM.nam w]
$ns_ use-newtrace

$ns_ trace-all $traceFile
$ns_ namtrace-all-wireless $namtraceFile $val(x) $val(y)

# SETUP TOPOGRAPHY

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

set god_ [create-god $val(nn)]                  ;# Global information

set chan_1_ [new $val(chan)]

# Communication Range setup

Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.00435e-08    ;# 3.00435e-08 for 80 meters)
Phy/WirelessPhy set Rb_ 2*1e6
Phy/WirelessPhy set Pt_ 0.281838   ; #0.281838 <- VERY IMPORTANT
Phy/WirelessPhy set freq_ 9.14e+08
Phy/WirelessPhy set L_ 1.0

;#  Set RXThresh_ 1.20174e-07 for 40 m community mobility model
;# 3.00435 e-08 for 80m for RPGM;
```

```
;# and 8.91754e-10 for 200 m as for RandomWayPoint and RandomWalk

# Configure nodes
        $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 \
                          -channel $chan_1_ \
                          -agentTrace ON \
                          -routerTrace ON \
                          -macTrace OFF \
                          -movementTrace OFF \
                          -energyModel $val(energyMdl) \
                          -txPower $val(TxPwr) \
                          -initialEnergy $val(initenergy)

# Create nodes

for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node ]
                $node_($i) random-motion 0       ;#0 means disable random motion
        }

#Define traffic pattern

source $val(cp)

# Define node mobility pattern declared above

source $val(sc)

proc finish {} {
    global  ns_ traceFile namtraceFile val
    $ns_ at $val(stop) "$ns_ nam-end-wireless $val(stop)"
    $ns_ flush-trace
    close $traceFile
    close $namtraceFile
    puts "SIMULATION OVER"
    $ns_ halt
 #   exec nam normalNAM.nam &
    exit 0
}

$ns_ at $val(stop) "finish"

$ns_ run
```

Appendix B

Linux Scripting commands Used for scenario generation and data gathering

Community_model parameters

./movGen -n 100 -t 500.0 -r 500.0 -s 0.0 -S 5.0 -p 0.7 -X 500.0 -Y 500.0 -R 3-C 3 -T 40.0 -w 0.3 -G 1 -g 0.0 -c 10 -v 3.0 -a on -d off

Referrence point group mobility model parameters

./rpgm 1 100 10.0 1.5 500.0 500.0 500.0 3.5 1.0 20.0 5.0 N

Random walk mobility model parameters used

./randwalk 100 500.0 500.0 500.0 3.5 1.0 30.0 N

Random way point mobility model parameters used

./mobgen 100 500.0 500.0 500.0 3.5 1.0 20.0 5.0 N

- Unix utilites used for data gathering and analysis

```
grep "PS_ID 4" normal.tr |grep "Ni 117" |grep "^s"| grep "Nl RTR">reply_binding
grep "PS_ID 5" normal.tr| grep "Pd 117" |grep "^s" |grep "Nl RTR">regstr
grep "PS_ID 4" normal.tr |grep "Pd 25"| grep "Nl RTR" |grep "^s"> reqst_binding
grep "Pd -1" normal.tr |grep "^s" |grep "Nl AGT">send_bdcst
grep "It cbr" normal.tr | grep "Ni 131"|grep "^r">cbr_traffic_131
```

(For the CBR thruput analysis AT AGENT LAYER!)

```
grep "It cbr" normal.tr |grep "Is 55" |grep "Ni 6" |grep "^r">cbr_traffic_6
```

(AT AGENT LAYER CBR thruput)

```
grep "$node_(0)" rpgm_100 >node_0

cut -d" " -f 6,7 node_0 >rpgm_node_0
cut -d" " -f 3 receive>response_time
awk -f Counter.awk receive
grep "^r" normal.tr |grep "PS_ID 3"> receive
grep "^s" normal.tr |grep "PS_ID 4" |grep "Ni 41" |grep "Nl AGT">receive
grep "^s" normal.tr |grep "Ni 41" |grep "PS_ID 4" >transmit.tr
```

(For filtering out directory node responses to requests)

```
grep "Ni 38" normal.tr |grep "PS_ID 3" >receive.tr.
```

(For filtering a client node receiving a service response)

```
grep "^s" normal.tr |grep "Ni 41" |grep "PS_ID 4" >receive.tr
```

Appendix C

Perl Script for Computing throughput, end-to-end delay, and PDR

```
 $sum = 0;
 $rx_pid = 0;
 $tx_pid = 0;
 $count = 0;
 $pd_start = argv[0];  # Enter the beginning of the observation
period
 $clock = $pd_start; # Time when observation begins (Period start)
 $pd_end = argv[1]; # Enter the end of observation period
 $granularity = 10; # Every ten seconds compute the throughput
 $agg_count = 0;
 $agg_thruput = 0;
 $thruput_sum = 0;

 open (receive_file,"cbr_traffic_10")   # Enter here the CBR client file
          || die ("Can't open $rxfile");

   while(<receive_file>) {

      @rx = split(' ');
      $rx_pid = $rx[46];
      print ("I read rx_pid =  ", $rx_pid, "\n"); # for debugging

      open (send_file,"cbr_traffic_70")     # Enter here the CBR source file
           || die ("Can't open $txfile");

     while (<send_file>)
     {
        @tx = split(' ');
        $tx_pid = $tx[46];
        print("Read tx_pid = ", $tx_pid, "\n");
        if ($tx_pid == $rx_pid)
        {
         if(($rx[2] - $tx[2]) < 10)
         {
           $end_time = $rx[2]; # identifies time of the last valid pkt
           $sum = $sum + ($rx[2] - $tx[2]);
           $count = $count + 1;

           if(($rx[2] - $clock) <= $granularity)
                {
                  $thruput_sum = $thruput_sum + $rx[36]; # Sum of bytes
                }
```

```
        else
             {
               $thruput = ($thruput_sum/$granularity)*(8/1024); # Kbps
               $agg_thruput = $agg_thruput + $thruput;
               $agg_count = $agg_count + 1;
               $clock = $clock + $granularity;

               while (($rx[2] - $clock) > $granularity)
                   {
                     $agg_count = $agg_count + 1;
                     $clock = $clock + $granularity;
                   }
               $thruput_sum = $rx[36];

             }
         }
        $tx_pid = 0;
        close(send_file);
    }
  } # end of while <send_file>

 } # end of while <receive_file>

if((309.1 - $end_time) > 0)
  {
    $thruput = ($thruput_sum/$granularity)*(8/1024); # Kbps
    $agg_thruput = $agg_thruput + $thruput;
    $agg_count = $agg_count +1;
  }

close (receive_file);
$delay = $sum/$count;
$avg_thruput = $agg_thruput/10; # There're 10 groups of time intervals
                                  out of 100.0 sec Observation!
print ("End time:= ",$end_time, ", Clock:= ",$clock, "\n");

print ("Total  packets received:  = ", $count, "\n");
print ("Total aggregate count:= ",$agg_count,"\n");
print("Average end-to-end delay= ", $delay, "sec","\n");
print (" Average throughput:= ", $avg_thruput," Kbps", "\n");
exit(0);
```