



P-1640



**IMPLEMENTATION OF AD-HOC ON-DEMAND  
POSITION BASED PRIVATE ROUTING PROTOCOL**

**A PROJECT REPORT**

*Submitted by*

<b>ARUN.V.N</b>	<b>71202205004</b>
<b>BABU KUMAR.R</b>	<b>71202205005</b>
<b>SAMPATH KUMAR.S</b>	<b>71202205038</b>

*in partial fulfillment for the award of the degree*

*Of*

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**

**KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2006**

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**IMPLEMENTATION OF AD-HOC ON-DEMAND POSITION BASED PRIVATE ROUTING PROTOCOL**” is the bonafide work of “**ARUN.V.N , BABU KUMAR.R and SAMPATH KUMAR.S** ” who carried out the project work under my supervision.

  
**SIGNATURE**

DR.G.GOPALSAMY, Ph.D


**HEAD OF THE DEPARTMENT**

Information Technology

  
**SIGNATURE**

Prof.K.R.BASKARAN, B.E, M.S

**GUIDE**

  
**SIGNATURE** 24/09/2006

Mr.K.RAMASUBRAMANIAN, M.C.A, M.E

**GUIDE**

The candidates with University Register Nos.**71202205004**, **71202205005** and **71202205038** were examined by us in the project viva-voce examination held on *24.09.06*.

  
INTERNAL EXAMINER

  
EXTERNAL EXAMINER

## ACKNOWLEDGEMENT

We express our sincere thanks to our Chairman Arutchelvar **Dr.N.Mahalingam, B.Sc., F.I.E** and Correspondent **Dr.K.Arumugam, B.E., M.S., M.I.E.**, for all their support and ray of strengthening hope extended.

We are extremely grateful to **Dr.K.K.Padmanabhan, B.Sc(Engg), M.Tech, Ph.D**, Principal, and Kumaraguru College of Technology for having given us a golden opportunity to embark on this project.

We are deeply obliged to **Dr.G.Gopalasamy, Ph.D** Head of the Department of Information Technology for their valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our project co-coordinator and guide **Prof.K.R.Baskaran, B.E, M.S**, Assistant Professor, Department of Information Technology for providing us his support and guidance which really helped us.

We are indebted to our co-guide **Mr.K.Ramasubramanian, M.C.A, M.E**, Lecturer, Department of Information Technology, for his helpful guidance and valuable support given to us throughout this project.

We are also thankful to all the faculty members and lab technicians of the Department of Information Technology for providing us the technical support during the course of this project.

We express our humble gratitude and thanks to our parents and our friends who helped us to complete this project successfully.

## DECLARATION

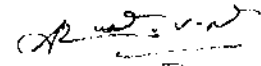
We,

<b>ARUN.V.N</b>	<b>71202205004</b>
<b>BABU KUMAR.R</b>	<b>71202205005</b>
<b>SAMPATH KUMAR.S</b>	<b>71202205038</b>

Declare that the project entitled “**IMPLEMENTATION OF AD-HOC ON-DEMAND POSITION BASED PRIVATE ROUTING PROTOCOL**”, submitted in partial fulfillment to Anna University as the project work of Bachelor Of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of Mr. K.R.Baskaran, M.S., Assistant Professor and Mr.K.Ramasubramanian, M.C.A, M.E, Lecturer, Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

Date: 24.04.06.

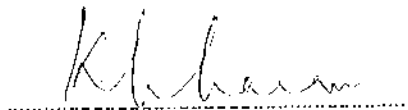


[Arun.V.N]

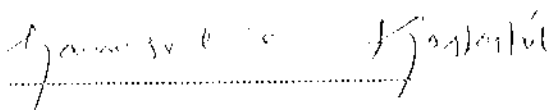
R. Babu, Kumari  
[Babu Kumar.R]

A. Sampath Kumar  
[Sampath Kumar.S]

Project Guided by



[Prof.K.R.Baskaran, B.E, M.S.,]



[Mr.K.Ramasubramanian, M.C.A, M.E.,]

## ABSTRACT

A mobile Ad-hoc network is an autonomous system of mobile nodes characterized by wireless links. The central challenge in the design of Ad-hoc networks is the development of the dynamic routing protocol that finds route between mobile nodes and provides more security during communication. Many such routing protocols have already been proposed which could provide security only to a certain extent such as AODV (Ad hoc On-demand Distance Vector Routing Protocol)

AODV is a routing protocol for the operation of ad-hoc networks. Each Mobile Host operates as a specialized router and routes are obtained as needed i.e. on-demand with little or no reliance on periodic advertisements. AODV provides loop free routes even while repairing broken links. Because the protocol does not require global periodic routing advertisements, the demand on the overall bandwidth available to the mobile nodes is substantially less than in those protocols that do necessitate such advertisements.

Here we present a new protocol called “**Ad hoc On-Demand Position-Based Private Routing Protocol (AO2P)**” for Ad-hoc networks which provides more secured routing and reliable communication between nodes. It maintains most of the advantages maintained in both AODV and several other Distance-Vector routing algorithms. The performance of this algorithm is evaluated in terms of qualitative metrics such as packet delivery ratio and secured data delivery.

# TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	<b>v</b>
	<b>LIST OF FIGURES</b>	<b>viii</b>
	<b>LIST OF TABLES</b>	<b>viii</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>viii</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Overview of Existing System	1
	1.1.1. Ad hoc On-demand Distance Vector (AODV)	1
	1.1.2 Drawbacks of existing system	2
	1.2 Overview of proposed system	2
	1.2.1 AO2P Routing Protocol	2
	1.2.2 Advantages	4
<b>2.</b>	<b>SYSTEM REQUIREMENT ANALYSIS</b>	<b>5</b>
	2.1 Product definition	5
	2.2 Project plan	6
<b>3.</b>	<b>SYSTEM REQUIREMENT SPECIFICATION</b>	<b>7</b>
	3.1 Purpose	7
	3.2 Scope	7
	3.3 Definition	7
	3.4 Abbreviations	8
	3.5 General description	8
	3.5.1 Product overview	8
	3.5.2 General assumptions	8
	3.6 Development and Operating environment	8
	3.6.1 Software requirements	8
	3.6.2 Hardware requirements	8

	3.7 Design constraints	9
	3.8 Software constraints	9
<b>4.</b>	<b>SYSTEM STUDY</b>	<b>10</b>
	4.1 AO2P	10
	4.2 PARSEC	12
	4.3 GloMoSim	13
	4.3.1 Architecture	14
	4.3.2 Directory structure of GloMoSim	15
	4.4 C	16
	4.4.1 Importance of C	16
	4.4.2 Why use C?	17
	4.4.3 Uses of C	18
	4.5 JAVA	19
	4.5.1 Introduction	19
	4.5.2 Features of Java	19
	4.5.3 Advantages of Java	20
<b>5.</b>	<b>DESIGN DOCUMENTS</b>	<b>21</b>
	5.1 Input Design	21
	5.2 Process Design	29
	5.2.1 Flow Chart	29
	5.2.2 Data Flow Diagram	33
<b>6.</b>	<b>TESTING</b>	<b>35</b>
	6.1 Product Testing	35
	6.2 System Testing	36
<b>7.</b>	<b>FUTURE ENHANCEMENTS</b>	<b>37</b>
<b>8.</b>	<b>CONCLUSION</b>	<b>38</b>
<b>9.</b>	<b>APPENDIX</b>	<b>39</b>
	9.1 Sample Source Code	39
	9.2 Sample Output	49
<b>10.</b>	<b>REFERENCES</b>	<b>54</b>

## LIST OF TABLES

## PAGE NO

1. Table 2.1	- Project plan	6
2. Table 4.1	- Models currently available in GloMoSim	15

## LIST OF FIGURES

1. Fig 4.1	- Message flow in AO2P routing discovery	11
2. Fig 4.2	- GloMoSim Architecture	14
3. Fig 5.1	- Flow chart for AO2P	29
4. Fig 5.2	- Flow chart for Position Management	30
5. Fig 5.3	- Flow chart for Get Position	31
6. Fig 5.4	- Flow chart for Tx & Rx	31
7. Fig 5.5	- Flow chart for Route Discovery	32
8. Fig 5.6	- Level 0 DFD for AO2P	33
9. Fig 5.7	- Level 1 DFD for AO2P	33
10. Fig 5.8	- Level 2 DFD for Position Management	34
11. Fig 5.9	- Level 2 DFD for Route Discovery	34

## LIST OF SYMBOLS

1. AO2P	- Ad-hoc On-demand Position based Private Routing Protocol
2. AODV	- Ad-hoc On-demand Distance Vector Routing Protocol
3. GPSR	- Greedy Perimeter Stateless Routing
4. GPS	- Global Positioning System
5. GLOMOSIM	- Global Mobile Simulator
6. PARSEC	- PARallel Simulation Environment for Complex systems



---

# *INTRODUCTION*

---

# 1. INTRODUCTION

An ad hoc wireless network is a dynamic network without any fixed infrastructure and centralized control system. Indeed all components (nodes) of this network are potentially mobile without any wired connection between them, and any node can act as a router. This freedom of mobility, however, results in frequent link disconnections between the components since they have a limited communicating distance determined by their radio range. Ad hoc wireless networks are normally used for specific strategic purposes, such as military deployment, emergency task force operation or temporary special business activity where it is not possible to have a fixed infrastructure. Due to the constantly varying topology in mobile ad hoc networks, it is quite difficult to maintain the entire network's routing information accurately and guarantee the message delivery. Therefore, dynamic multihop paths are constructed to route messages while mobile nodes cooperate wandering.

## 1.1 OVERVIEW OF EXISTING SYSTEM:

### 1.1.1. AD HOC ON-DEMAND DISTANCE VECTOR (AODV)

AODV is essentially a combination of DSR (Dynamic Source Routing) and DSDV (Highly Dynamic Destination Sequenced Distance Vector Routing). It borrows the basic on-demand mechanism of route Discovery and Route maintenance from DSR, plus the use of hop-by-hop the routing, sequence number and periodic beacon from DSDV. When a source *S* needs a path to some destination *D*, it broadcasts a route request message enclosing the last known sequence number to that destination. The route request is broadcasted across the network until it reaches a node that has a route to the destination with the destination sequence number higher than that enclosed in the request. Each node that forwards the route request creates a reverse route for itself back to node *S*, when the route request reaches a node with a route to *D*, that node generates a route reply that contains the number of hops necessary

to reach D and the sequence number for D most recently seen by the node generating the reply. Each node that participates in forwarding this reply back forward the originator of the route request (node S) creates a forward route to D. The state created in each node along the path from S to D is hop-by-hop state that is, each node remembers only the next hop and not the entire route as would be done in source routing.

In order to maintain routes, AODV (Ad-hoc On-demand Distance Vector Routing) normally requires that each node periodically transmit a HELLO message, with a default rate of once per second. Failure to receive three consecutive HELLO message from a neighbor is taken as an indication that the link to the neighbor in question is down. When a link goes down, any upstream node that has recently forward packets to destination using that link is notified via an unsolicited route reply containing an infinite metric for that destination. Upon receipt of such a route reply, a node must acquire a new route to the destination using route discovery.

### **1.1.2 DRAWBACKS OF EXISTING SYSTEM**

- Uses a single path which is unreliable.
- Low packet delivery ratio.
- No load balancing.
- Increased packet drop ratio.
- No proper security.

## **1.2 OVERVIEW OF PROPOSED SYSTEM:**

### **1.2.1 AO2P ROUTING PROTOCOL**

In AO2P, all the nodes first get its position from the GPS (Global Positioning System) enabled system. The position values are being communicated to the trusted server. The source gets the position of the destination from the server. When the source gets the position of its

destination, it also gets the time when the position is updated and an authentication code.

In AO2P (Ad-hoc On-demand Position based Private Routing Protocol), a source discovers the route through the delivery of a routing request to its destination. Once a source needs to find the route to its destination, it first generates a pseudo Identification (ID) and a temporary MAC (Message Authentication Code) address for itself through a globally defined hash function using its position and the current time as the inputs. The source then sends out a routing request (rreq) message. The rreq message carries the information needed for routing, such as the position of the destination and the distance from this source to the destination, as well as the source pseudo ID.

Since it is possible that another node has updated the same position (yet at a different time) to the position servers, a destination challenge message is carried in the rreq to make sure that the right destination will be reached. This message is also a result of a hash function, of which the inputs are the position of the destination and the time at which this position is updated. The neighboring nodes around the source, called receivers, will receive the rreq. A receiver checks the destination challenge message to find out whether it is the destination. If not, a receiver assigns itself to a receiver class. Each receiver uses a hash function to generate a pseudo ID and a temporary MAC address.

The receivers then contend for the wireless channel to send out a hop reply (hrep) message in a so-called rreq contention phase. The receiver who has successfully sent out the hrep will be the next hop. Its pseudo ID is carried in the hrep. On receiving the hrep, the source replies with a confirm (cnfm) message. Its next hop replies to this message with an ack. Upon receiving the ack, the source saves the pseudo ID and the temporary MAC address of the next hop in its routing table. After receiving the cnfm, the next-hop receiver becomes a sender. It sends out the modified rreq, which carries the distance from itself to

the destination. The TTL (Time-to-Live) value is reduced by 1. Neighboring nodes around it will contend to be its next hop. Once the sender receives a hrep, it couples the pseudo ID and the temporary MAC address of its next hop with those of its previous hop and saves the pairs in the routing table. The searching of the next hop is repeated until the destination receives the rreq. After identifying the destination challenge message, the destination sends out a hrep. After receiving the cnfm from its previous hop, the destination sends a routing reply (rrep) message through the reverse path to the source.

The source finds out whether it reaches the right destination by decrypting the information with the destination's public key and comparing the authentication code with the one it obtained through the position request. A route discovery failure will occur when a sender cannot find a legitimate next hop. The source will start a new route search based on the destination's most updated position after a back off time. After a route is built up, data packets are delivered following the pseudo ID and temporary MAC address pairs in the routing tables. Routing maintenance mechanisms in traditional ad hoc routing algorithms can be used for AO2P. When a route is broken, an error message will be sent back.

### 1.2.2 ADVANTAGES

- Uses a single path which is reliable.
- Low packet loss ratio.
- Low packet drop ratio.
- Secured transmission.

---

*SYSTEM REQUIREMENT  
ANALYSIS*

---

## **2. SYSTEM REQUIREMENTS ANALYSIS**

During the system analysis phase a thorough analysis of the system is carried out to find the various features that can be incorporated to make the system better than the existing ones. The following are the objectives of conducting such an analysis:

- Evaluation of the feasibility of developing the system
- Evaluation of the cost of development
- Preparation of the System Requirements Specification

During this phase a flow of the system was clearly understood. The system was divided into modules and clearly outlined. The programming language most suitable for the development of the system and the minimum hardware requirements for the system are also identified.

### **2.1 PRODUCT DEFINITION:**

**AO2P:** Ad-Hoc On-demand Position based private routing Protocol is for mobile Ad-hoc networks. Position request and Route requests are broadcasted for initial route formation. AO2P provide increased secure and reliable data communication. This ensures privacy during communication.

## 2.2 PROJECT PLAN:

<b>WORK</b>	<b>DURATION</b>
Feasibility Analysis	One week
Literature survey	Five weeks
Requirements gathering	Two weeks
Study and analysis between various protocols	Two weeks
Abstract preparation	One week
Technology familiarization	Two weeks
Study about AODV	One weeks
Implementation of AO2P	Five weeks

Table 2.1: Project Plan



---

*SYSTEM REQUIREMENT  
SPECIFICATION*

---

## **3. SOFTWARE REQUIREMENTS SPECIFICATION**

### **3.1 PURPOSE:**

The purpose of this document is to specify the requirements of our project “Implementation of Ad-Hoc on-demand position-based private routing protocol”. It describes the interfaces for the system. The document also bridges the communication gap between the customer and analyst.

### **3.2 SCOPE:**

SRS forms the basis of agreement between the client and the supplier and what the software product will do. It also provides a reference for the validation of the final project.

Any changes made to the SRS in the future will have to go through formal change approval process.

### **3.3 DEFINITION:**

- **Customer**

A person or organization, internal or external to the producing organization, who takes financial responsibility for the system. In a large system this may not be the end user. The customer is the ultimate recipient of the developed project and its artifacts.

- **User**

A person who uses the system that is developed.

- **Analyst**

The analyst, details the specification of the system's functionality by describing the requirements aspect and other supporting software requirements.

### **3.4 ABBREVIATIONS:**

- SRS : Software Requirement Specification  
AO2P : Ad-hoc On-demand Position based Routing.

### **3.5 GENERAL DESCRIPTION:**

#### **3.5.1 PRODUCT OVERVIEW**

This project aims to provide reliability and security in ad-hoc wireless network. It uses AO2P protocol. A server contains the position information of all the nodes. Each and every node gets the position of another node from the server and transmits data by broadcasting a route request message along with the position of the destination. Thus more security is imposed as communication is based on position of destination.

#### **3.5.2 GENERAL ASSUMPTIONS**

- All the nodes get the position using a GPS enabled system.
- The nodes update their position information to the server.
- The server is assumed to be a trusted one.

### **3.6 DEVELOPMENT AND OPERATING ENVIRONMENT:**

#### **3.6.1 SOFTWARE REQUIREMENTS**

- Operating System : Windows XP  
Language : C, Java  
IDE : Glomosim 2.03 (Simulator)  
Compiler : PARSEC

#### **3.6.2 HARDWARE REQUIREMENTS**

- Processor : Intel Pentium IV  
RAM : 256 MB  
Hard disk : At least 2 GB free space

### **3.7 DESIGN CONSTRAINTS:**

The following assumptions are made while simulating the project

- The position of the nodes is specified in the input file.
- Source, destination and server nodes are specified in input file.

### **3.8 SOFTWARE CONSTRAINTS:**

The system requires that Glomosim 2.03 simulator, Java and C to be installed.

---

---

*SYSTEM STUDY*

---

---

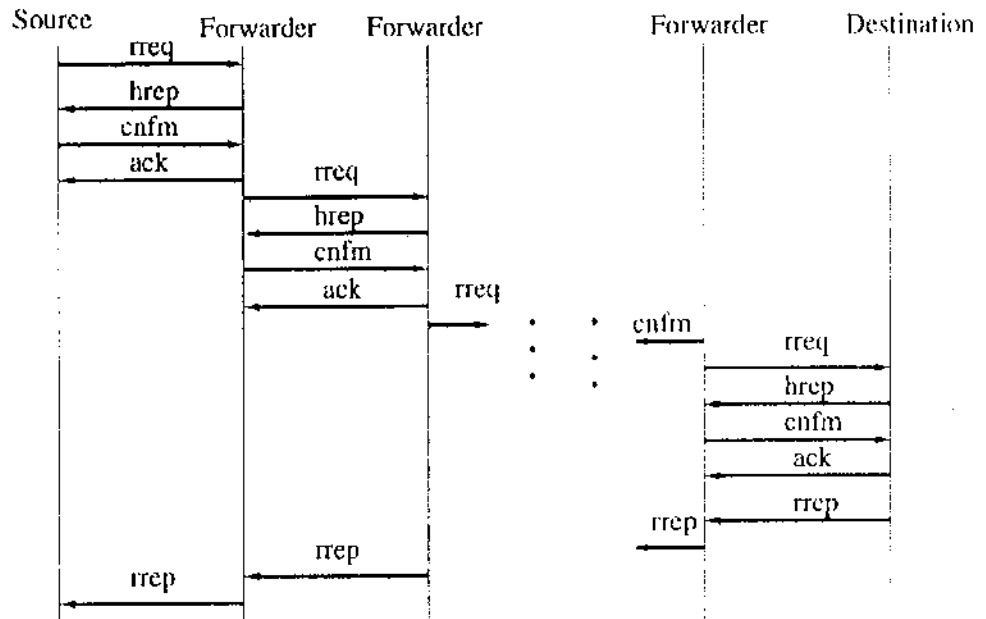
## 4. SYSTEM STUDY

### 4.1 AO2P:

In AO2P, a source discovers the route through the delivery of a routing request to its destination. A node en-route will generate a pseudo node ID and a temporary MAC address. Once a route is built up, data is forwarded from the source to the destination based on the pseudo IDs. Once a source needs to find the route to its destination, it first generates a pseudo ID and a temporary MAC address for itself through a globally defined hash function using its position and the current time as the inputs. Such a procedure makes the probability that two active nodes (i.e., nodes involved in routing) have the same ID and MAC address small and negligible. The source then sends out a routing request (rreq) message. The rreq message carries the information needed for routing, such as the position of the destination and the distance from this source to the destination, as well as the source pseudo ID.

Since it is possible that another node has updated the same position (yet at a different time) to the position servers, a destination challenge message is carried in the rreq to make sure that the right destination will be reached. This message is also a result of a hash function, of which the inputs are the position of the destination and the time at which this position is updated. Rreq carries the challenge message instead of the time for less information revelation. Rreq also carries a Time-to-Live (TTL) number that deals with the possible loop. TTL is the maximum number of the hops a rreq can be forwarded. A source node can estimate the TTL value according to the distance from the source to the destination and the radio transmission range for each hop. The neighboring nodes around the source, called receivers, will receive the rreq. A receiver checks the destination challenge message to find out whether it is the destination. If not, a receiver assigns itself to a

receiver class. Each receiver uses a hash function to generate a pseudo ID and a temporary MAC address.



**Fig 4.1 Message flow in AO2P routing discovery**

The inputs of the hash function are the receiver’s position and the time it receives the rreq. The receivers then contend for the wireless channel to send out a hop reply (hrep) message in a so-called rreq contention phase. The receiver who has successfully sent out the hrep will be the next hop. Its pseudo ID is carried in the hrep. On receiving the hrep, the source replies with a confirm (cnfm) message. Its next hop replies to this message with an ack. Upon receiving the ack, the source saves the pseudo ID and the temporary MAC address of the next hop in its routing table. After receiving the cnfm, the next-hop receiver becomes a sender. It sends out the modified rreq, which carries the distance from itself to the destination. The TTL value is reduced by 1. Neighboring nodes around it will contend to be its next hop. Once the sender receives a hrep, it couples the pseudo ID and the temporary MAC

address of its next hop with those of its previous hop and saves the pairs in the routing table. The searching of the next hop is repeated until the destination receives the rreq. After identifying the destination challenge message, the destination sends out a hrep. After receiving the cnfm from its previous hop, the destination sends a routing reply (rrep) message through the reverse path to the source.

The source finds out whether it reaches the right destination by decrypting the information with the destination's public key and comparing the authentication code with the one it obtained through the position request. A route discovery failure will occur when a sender cannot find a legitimate next hop. The source will start a new route search based on the destination's most updated position after a back off time. After a route is built up, data packets are delivered following the pseudo ID and temporary MAC address pairs in the routing tables. Routing maintenance mechanisms in traditional ad hoc routing algorithms can be used for AO2P. When a route is broken, an error message will be sent back to the source by the node that has discovered the broken link. In AO2P, during the communication, the destination will update its new position to the source through the reverse route. The source can thus start a new routing discovery using the updated position information.

## **4.2 PARSEC:**

PARSEC (for **PAR**allel Simulation Environment for Complex systems) is a C-based discrete-event simulation language. It adopts the process interaction approach to discrete-event simulation. An object (also referred to as a physical process) or set of objects in the physical system is represented by a logical process. Interactions among physical processes are modeled by time stamped message exchanges among the corresponding logical processes.



One of the important distinguishing features of PARSEC is its ability to execute a discrete-event simulation model using several different asynchronous parallel simulation protocols on a variety of parallel architectures. PARSEC is designed to cleanly separate the description of a simulation model from the underlying simulation protocol, sequential or parallel, used to execute. Thus with few modifications, a PARSEC program may be executed using the traditional sequential simulation protocol or one of many parallel optimistic or conservative protocols.

The PARSEC language is derived from Maisie, but with several improvements, both in syntax of the language and in its execution environment.

#### **4.3 GLOMOSIM:**

GLOMOSIM is a library-based sequential and parallel simulator of wireless networks; it is designed as a set of library modules, each of which simulates a specific wireless communication protocol in the protocol stack. The library has been developed using PARSEC: a C-based parallel simulation language. New protocols and modules can be programmed and added to the library using this language. GloMoSim has been designed to be extensible and composable.

GloMoSim (for Global Mobile Information System Simulator) effectively utilizes parallel execution to reduce the simulation time of detailed high-fidelity models of large communication networks.

### 4.3.1 ARCHITECTURE

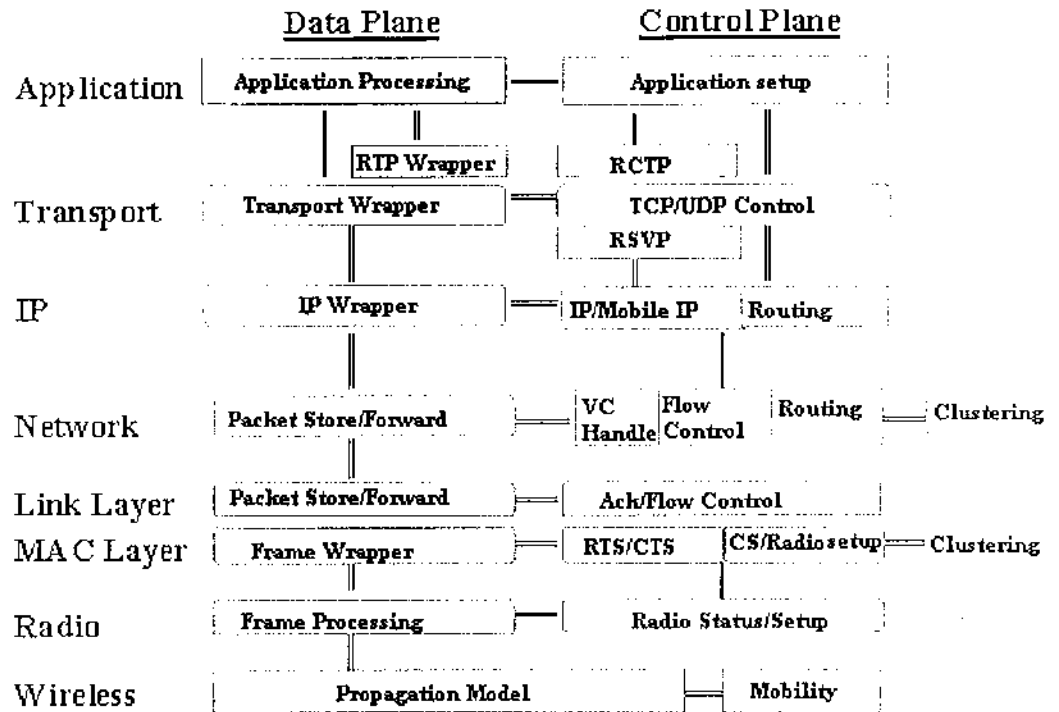


Fig 4.2 GloMoSim Architecture

The networking stack is decomposed into a number of layers. A number of protocols have been developed at each layer and models of these protocols or layers can be developed at different levels of granularity.

In GloMoSim, the simulation can be scaled to thousands of nodes. If we have to instantiate an entity for each node in the runtime, the memory requirements would increase dramatically. The performance of the system will also degrade rapidly. To circumvent these problems network gridding was introduced into the simulation.

The network gridding technique means that we can increase the number of nodes in the system while maintaining the same number of entities in the simulation. Some of the existing available GloMoSim models are shown below.

LAYER	MODELS
APPLICATION	TCPLIB(telnet,ftp), CBR(constant bit rate traffic), Replicated file system, HTTP
TRANSPORT	TCP(FreeBSD),UDP,NS TCP(Tahoe) and others
UNICAST ROUTING	AODV, Bellman-Ford, DSR. Fisheye, flooding, LAR(Scheme-1). NSDSDV, OSPF, WRP
MAC LAYER	CSMA, FAMA , MACA. IEEE 802.11
RADIO	Radio with and without capture capability
PROPAGATION	Freespace, Rayleigh, Ricean, SIR CIM
MOBILITY	Random waypoint, random drunken. ECRV, Group mobility

**Table 4.1 Models currently available in GloMoSim**

#### 4.3.2 DIRECTORY STRUCTURE OF GLOMOSIM

After downloading and unzipping GloMoSim, it should contain the following directories:

**\application** contains code for the application layer

**\bin** for executable and input and output files

**\include** contains common include files

**\mac** contains the code for the MAC layer

**\main** contains the basic framework design

**\network** contains the code for the network layer

**\radio** contains the code for the physical layer

**\transport** contains the code for transport layer

#### **4.4 C:**

C was evolved from ALGOL, BCPL and B by Dennis Ritchie at Bell Laboratories in 1972. C uses many concepts from these languages and added the concept of data types and other powerful features. Since it was developed along with UNIX operating system, it is strongly associated with UNIX. This OS which was also developed in bell laboratories was coded almost entirely in C.

The rapid growth of C led to development of different versions of the language that were similar but often incompatible. This posed a serious problem for system developers. To assure that the C language remains standard, in 1983, American National Standards Institute (ANSI) appointed a technical committee to define a standard for C. The committee approved a version of C in 1989 which is now known as ANSI C.

##### **4.4.1 IMPORTANCE OF C**

C is a robust language whose rich set of built-in functions and operators can be used to write any complex program. The C compiler combines the capabilities of an assembly language with the features of the high-level language and therefore it is suitable for writing both system software and business packages.

Programs written in C are efficient and fast. This is due to its variety of data types and powerful operators. It is many times faster than BASIC. There are only 32 keywords and its strength lies in its built-in functions. Several standard functions are available which can be used for developing programs.

C is highly portable. This means that C programs written for one computer can be run on another with little or no

modification. Portability is important if we plan to use a new computer with different OS.

C language is well suited for structured programming, thus requiring the user to think of a problem in terms of function modules or blocks. A proper collection of these modules would make a complete program. This modular structure makes program debugging, testing and maintenance easier.

#### 4.4.2 WHY USE C?

C has been used successfully for every type of programming problem imaginable from operating systems to spreadsheets to expert systems - and efficient compilers are available for machines ranging in power from the Apple Macintosh to the Cray supercomputers. The largest measure of C's success seems to be based on purely practical considerations:

- the portability of the compiler;
- the standard library concept;
- a powerful and varied repertoire of operators;
- an elegant syntax;
- ready access to the hardware when needed;
- and the ease with which applications can be optimized by hand-coding isolated procedures

C is often called a "Middle Level" programming language. This is not a reflection on its lack of programming power but more a reflection on its capability to access the system's low level functions. Most high-level languages (e.g. FORTRAN) provide everything the programmer might want to do already build into the language. A low level language (e.g. assembler) provides nothing other than access to the machines basic instruction set. A

middle level language, such as C, probably doesn't supply all the constructs found in high-languages - but it provides you with all the building blocks that you will need to produce the results you want!

#### 4.4.3 USES OF C

C was initially used for system development work, in particular the programs that make-up the operating system. Why use C? Mainly because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

- Operating Systems
- Language Compilers
- Assemblers
- Text Editors
- Print Spoolers
- Network Drivers
- Modern Programs
- Data Bases
- Language Interpreters
- Utilities

In recent years C has been used as a general-purpose language because of its popularity with programmers. It is not the world's easiest language to learn and you will certainly benefit if you are not learning C as your first programming language! C is trendy (I nearly said sexy) - many well established programmers are switching to C for all sorts of reasons, but mainly because of the portability that writing standard C programs can offer.

## 4.5. JAVA:

### 4.5.1. INTRODUCTION

Java is an object-oriented programming language with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets. Because of its rich set of API's, similar to Macintosh and windows, and its platform independence, Java can also be thought of as a platform in itself.

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer.

The Java platform has two components:

- The Java Virtual Machine
- The Java Application Programming Interface (API)

Java Virtual Machine is the base for the Java platform and is ported onto various hardware-based platforms. The API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. It is grouped into libraries of related classes and interfaces; these libraries are known as packages.

### 4.5.2. FEATURES OF JAVA

**Development Tools:** The development tools provide everything you'll need for compiling, running, monitoring,

---

*DESIGN  
DOCUMENTS*

---



## 5. DESIGN DOCUMENTS

### 5.1 INPUT DESIGN:

There are several modules available in each layer of GloMoSim. For our project the changes made in some modules are as follows:

\Glomosim\application\Application.pc:

.....

```
Else if(strcmp(buf,"AO2P")==0){  
}
```

.....

\Glomosim\network\nwip.pc:

```
#include "ao2p.h"
```

.....

```
case IPPROTO_AO2P: {
```

```
RoutingAo2pHandleProtocolPacket(node, msg, sourceAddress,  
destinationAddress, ttl);
```

```
break;
```

```
}
```

.....

```
case ROUTING_PROTOCOL_AO2P: {
```

```
RoutingAo2pHandleProtocolEvent(node, msg):
```

```
break;
```

```
}
```

.....

```
else if (strcmp(protocolString, "AO2P") == 0) {
```

```
ipLayer->routingProtocolChoice = ROUTING_PROTOCOL_AO2P;
```

```
RoutingAo2pInit( node, (GlomoRoutingAo2p**)&ipLayer->  
>routingProtocol, nodeInput);
```

```
}
```

.....

```
case ROUTING_PROTOCOL_AO2P:
```

```
    RoutingAo2pFinalize(node);
```

```
    break;
```

```
.....
```

```
\Glomosim\include\nwcommon.h:
```

```
#define IPPROTO_AO2P 123
```

```
\Glomosim\include\network.h:
```

```
typedef enum {
```

```
ROUTING_PROTOCOL_AO2P,
```

```
} NetworkRoutingProtocolType;
```

```
\Glomosim\main\makefile.pc
```

```
.....
```

```
..\network\ao2p.h
```

```
.....
```

```
..\network\ao2p.pc
```

```
\Glomosim\main\makent.pc
```

```
.....
```

```
call pcc %parsecflags% -I..\include\ -I..\network\ -clock longlong -c
```

```
..\network\ao2p.pc
```

```
.....
```

```
\Glomosim\bin\config.in
```

```
# The following parameter represents the maximum simulation time.
```

```
The #numbered
```

```
# Portion can be followed by optional letters to modify the simulation  
time.
```

```
# for example:
```

```
#      100NS    - 100 nano-seconds
```

```
# 100MS - 100 milli-seconds
# 100S - 100 seconds
# 100 - 100 seconds (default case)
# 100M - 100 minutes
# 100H - 100 hours
# 100D - 100 days
SIMULATION-TIME 200S
```

```
# The following is a random number seed used to initialize part of the
seed of
# various randomly generated numbers in the simulation. This can be
used to vary
# the seed of the simulation to see the consistency of the results of the
# simulation.
```

```
SEED 1
```

```
# The following two parameters stand for the physical terrain in which
the nodes
# are being simulated. For example, the following represents an area of
size 100
# meters by 100 meters. All range parameters are in terms of meters.
TERRAIN-DIMENSIONS (700, 700)
```

```
# the following parameter represents the number of nodes being
simulated.
```

```
NUMBER-OF-NODES 10
```

```
# The following parameter represents the node placement strategy.
-- RANDOM: Nodes are placed randomly within the physical terrain.
-- UNIFORM: Based on the number of nodes in the simulation, the
physical
```

```
# terrain is divided into a number of cells. Within each cell, a node is
# placed randomly.
#- GRID: Node placement starts at (0, 0) and are placed in grid format
with
# each node GRID-UNIT away from its neighbors. The number of
nodes has to be
# square of an integer.
#- FILE: Position of nodes is read from NODE-PLACEMENT-FILE.
On each line of
# the file, the x and y position of a single node is separated by a space.
```

```
NODE-PLACEMENT    FILE
NODE-PLACEMENT-FILE ./nodes.input
# NODE-PLACEMENT    GRID
# GRID-UNIT        30
#NODE-PLACEMENT    RANDOM
# NODE-PLACEMENT    UNIFORM
```

```
# the following represents parameters for mobility. If MOBILITY is set
to NO,
```

```
# than there is no movement of nodes in the model.
```

```
MOBILITY NONE;
```

```
# Random Waypoint and its required parameters.
```

```
#MOBILITY RANDOM-WAYPOINT
#MOBILITY-WP-PAUSE    100S
#MOBILITY-WP-MIN-SPEED  0
#MOBILITY-WP-MAX-SPEED  20
#MOBILITY TRACE
#MOBILITY-TRACE-FILE ./mobility.in
```

```

#MOBILITY PATHLOSS-MATRIX
# The following parameters are necessary for all the mobility models

MOBILITY-POSITION-GRANULARITY 0.3333

# PROPAGATION-LIMIT:
# Signals with powers below PROPAGATION-LIMIT (in dBm)
# are not delivered.
PROPAGATION-LIMIT -111.0

# PROPAGATION-PATHLOSS: path loss model
# FREE-SPACE:
# Friss free space model.
# (path loss exponent, sigma) = (2.0, 0.0)
# TWO-RAY:
# Two ray model. It uses free space path loss
# (2.0, 0.0) for near sight and plane earth
# path loss (4.0, 0.0) for far sight. The antenna
# height is hard-coded in the model (1.5m).
# PATHLOSS-MATRIX:

#PROPAGATION-PATHLOSS FREE-SPACE
PROPAGATION-PATHLOSS TWO-RAY
#PROPAGATION-PATHLOSS PATHLOSS-MATRIX
#PROPAGATION-FADING-MODEL RAYLEIGH
#PROPAGATION-FADING-MODEL RICIAN
RICIAN-K-FACTOR 5

# NOISE-FIGURE: noise figure
NOISE-FIGURE 10.0

```

```

# TEMPERATURE: temperature of the environment (in K)
TEMPERATURE 290.0

# RADIO-TYPE: radio model to transmit and receive packets
# RADIO-ACCNOISE: standard radio model
RADIO-TYPE RADIO-ACCNOISE

# RADIO-FREQUENCY: frequency (in hertz) (Identifying variable for
multiple
# radios)
RADIO-FREQUENCY 2.4e9
# RADIO-BANDWIDTH: bandwidth (in bits per second)
RADIO-BANDWIDTH 2000000
# RADIO-RX-TYPE: packet reception model
# SNR-BOUNDED:
# If the Signal to Noise Ratio (SNR) is more than
# RADIO-RX-SNR-THRESHOLD (in dB), it receives the signal
# without error.
RADIO-RX-TYPE SNR-BOUNDED
RADIO-RX-SNR-THRESHOLD 9.1
#RADIO-RX-TYPE BER-BASED
#BER-TABLE-FILE ./ber_dbpsk.in
# RADIO-TX-POWER: radio transmission power (in dBm)
RADIO-TX-POWER 15.0
# RADIO-ANTENNA-GAIN: antenna gain (in dB)
RADIO-ANTENNA-GAIN 0.0
# RADIO-RX-SENSITIVITY: sensitivity of the radio (in dBm)
RADIO-RX-SENSITIVITY -91.0
# RADIO-RX-THRESHOLD: Minimum power for received packet (in
dBm)
RADIO-RX-THRESHOLD -81.0

```

```

MAC-PROTOCOL      802.11
#MAC-PROTOCOL     CSMA
#MAC-PROTOCOL     MACA
#MAC-PROTOCOL     TSMA
#TSMA-MAX-NODE-DEGREE  8
#MAC-PROPAGATION-DELAY 1000NS

NETWORK-PROTOCOL  IP
NETWORK-OUTPUT-QUEUE-SIZE-PER-PRIORITY 100

#RED-MIN-QUEUE-THRESHOLD 150
#RED-MAX-QUEUE-THRESHOLD 200
#RED-MAX-MARKING-PROBABILITY 0.1
#RED-QUEUE-WEIGHT .0001
#RED-TYPICAL-PACKET-TRANSMISSION-TIME 64000NS

#ROUTING-PROTOCOL  BELLMANFORD
ROUTING-PROTOCOL  AO2P
#ROUTING-PROTOCOL  DSR
#ROUTING-PROTOCOL  LAR1
#ROUTING-PROTOCOL  WRP
#ROUTING-PROTOCOL  FISHEYE
#ROUTING-PROTOCOL  ZRP
#ZONE-RADIUS      2
#ROUTING-PROTOCOL  STATIC
#STATIC-ROUTE-FILE  ROUTES.IN

```

```

# The following is used to setup applications such as FTP and Telnet.
# The file will need to contain parameters that will be use to
# determine connections and other characteristics of the particular
# application.

```

```

APP-CONFIG-FILE ./cbr.in
# The following parameters determine if you are interested in the
statistics of
# a a single or multiple layer. By specifying the following parameters as
YES,
# the simulation will provide you with statistics for that particular layer.
All
# the statistics are compiled together into a file called "GLOMO.STAT"
that is
# produced at the end of the simulation. APPLICATION-STATISTICS
YES
TCP-STATISTICS          NO
UDP-STATISTICS          YES
ROUTING-STATISTICS      YES
NETWORK-LAYER-STATISTICS  YES
MAC-LAYER-STATISTICS     YES
RADIO-LAYER-STATISTICS   YES
CHANNEL-LAYER-STATISTICS  YES
MOBILITY-STATISTICS      YES

# GUI-OPTION: YES allows GloMoSim to communicate with the Java
Gui Vis Tool
#          NO does not

GUI-OPTION  YES
GUI-RADIO   YES
GUI-ROUTING YES

```



## 5.2 PROCESS DESIGN:

### 5.2.1 FLOW CHART

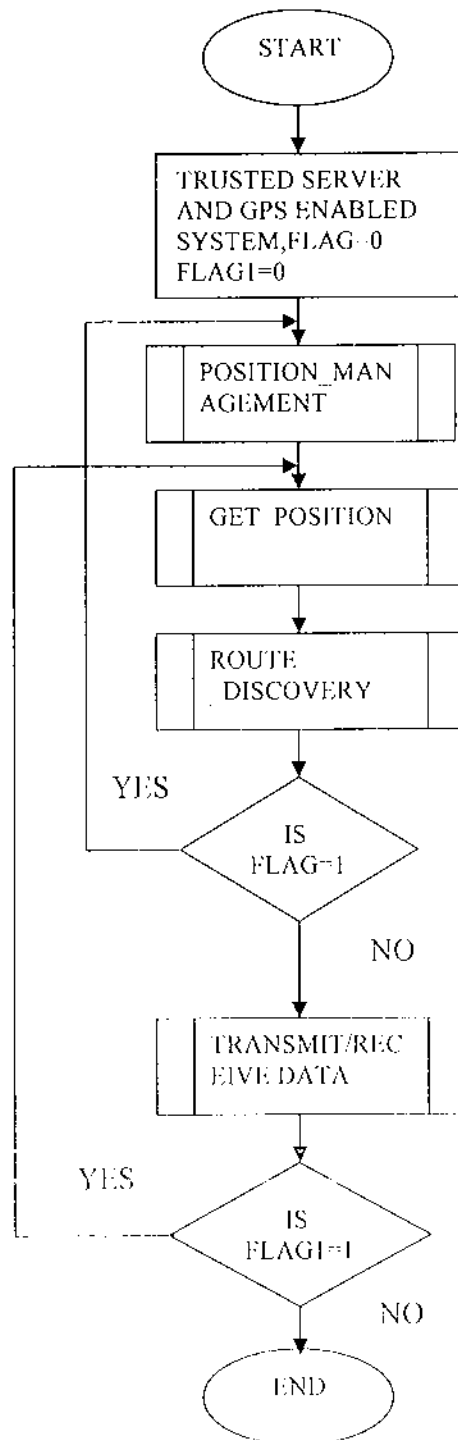


Fig 5.1 Flow chart for AO2P

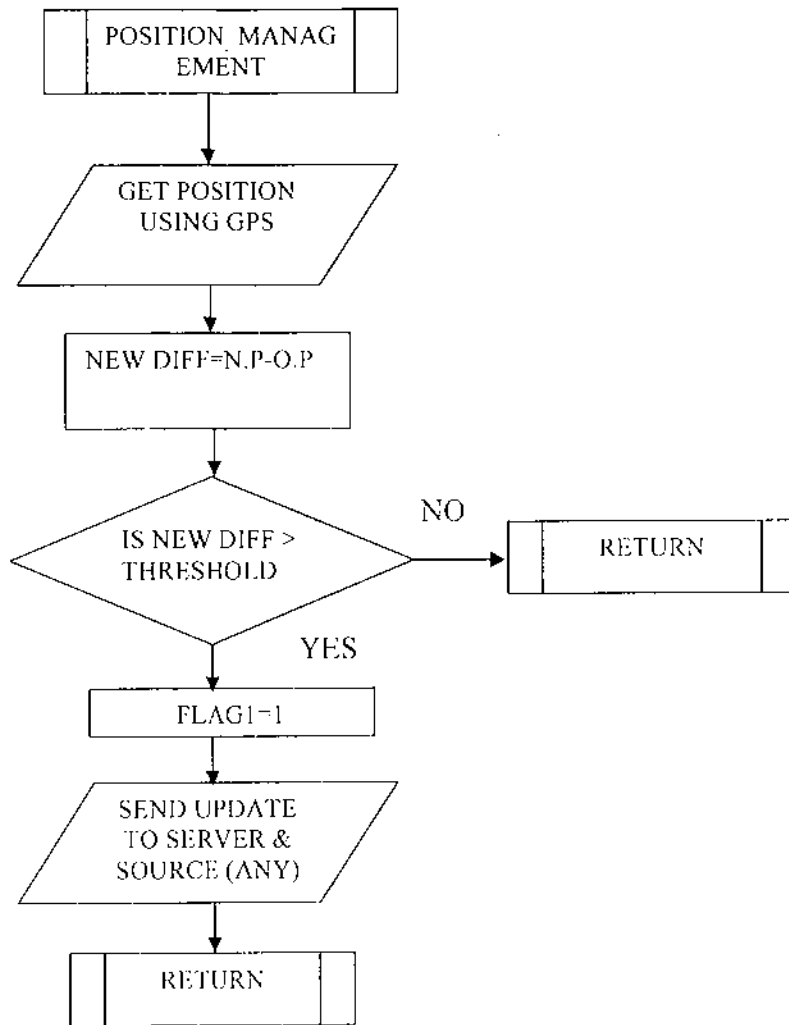


Fig 5.2 Flow chart for position management

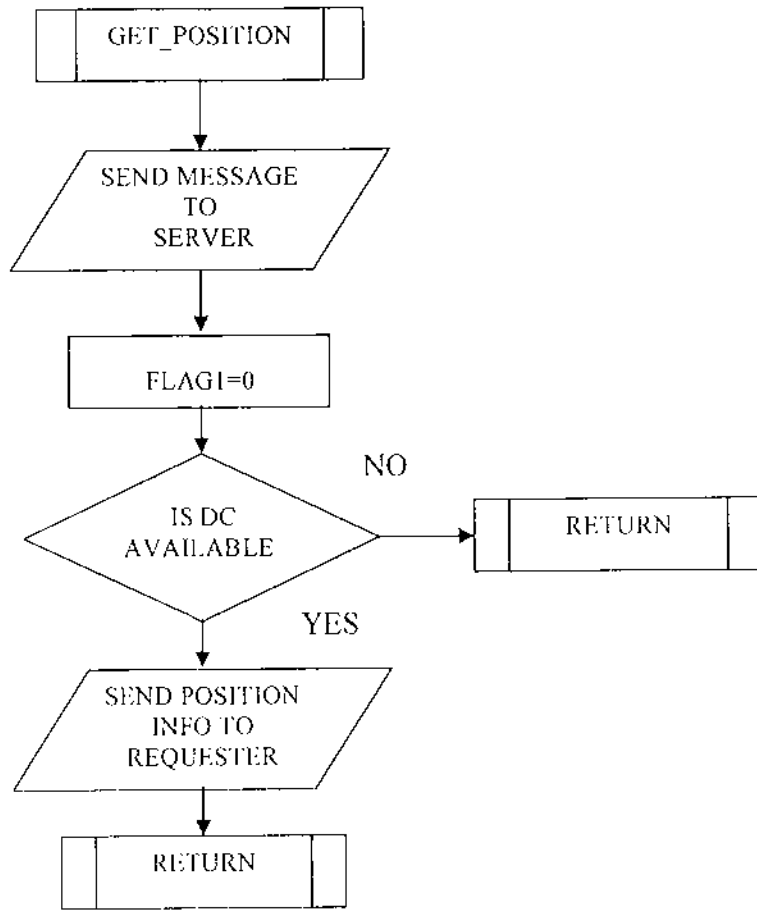


Fig 5.3 Flow chart for get position.

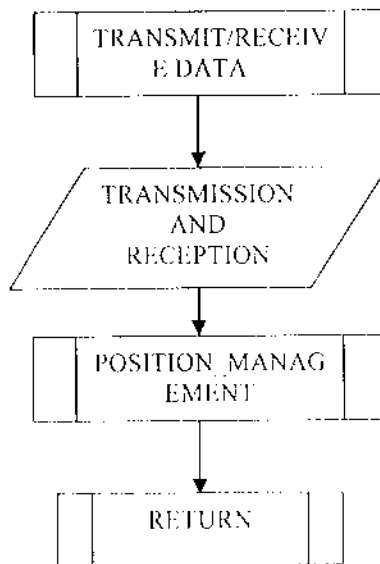


Fig 5.4 Flow chart for Tx & Rx

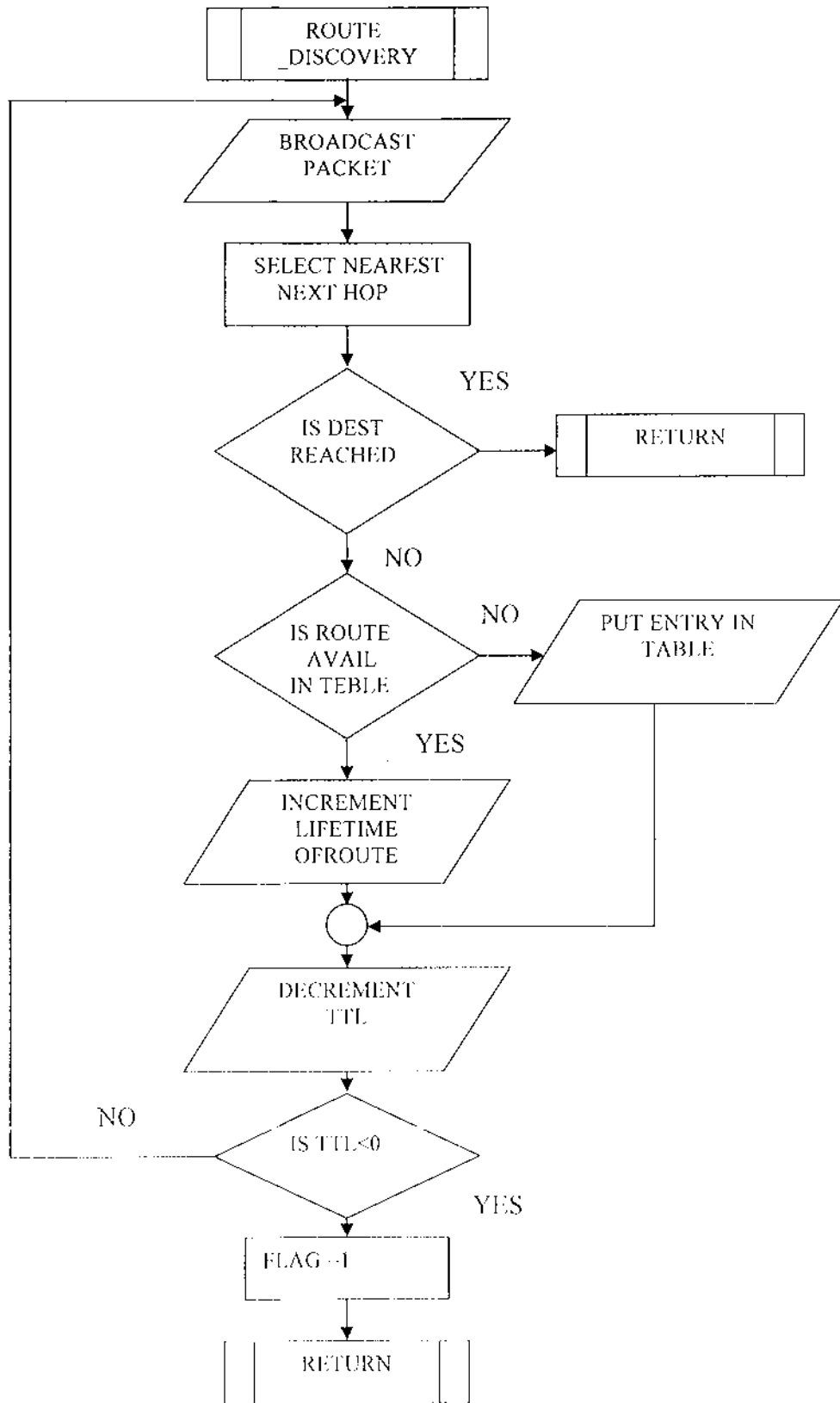


Fig 5.5 Flow chart for Route Discovery

## 5.2.2 DATA FLOW DIAGRAM

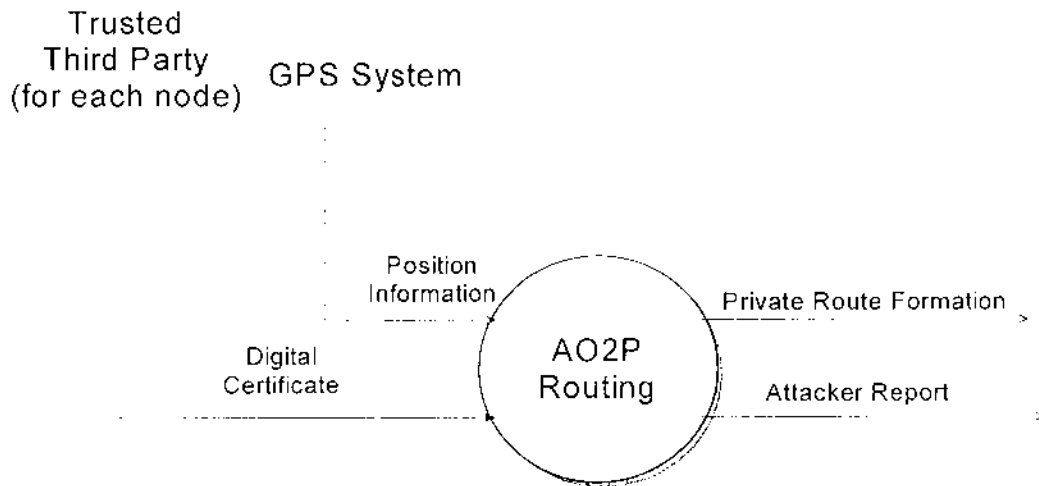


Fig 5.6 Level 0 DFD for AO2P Routing

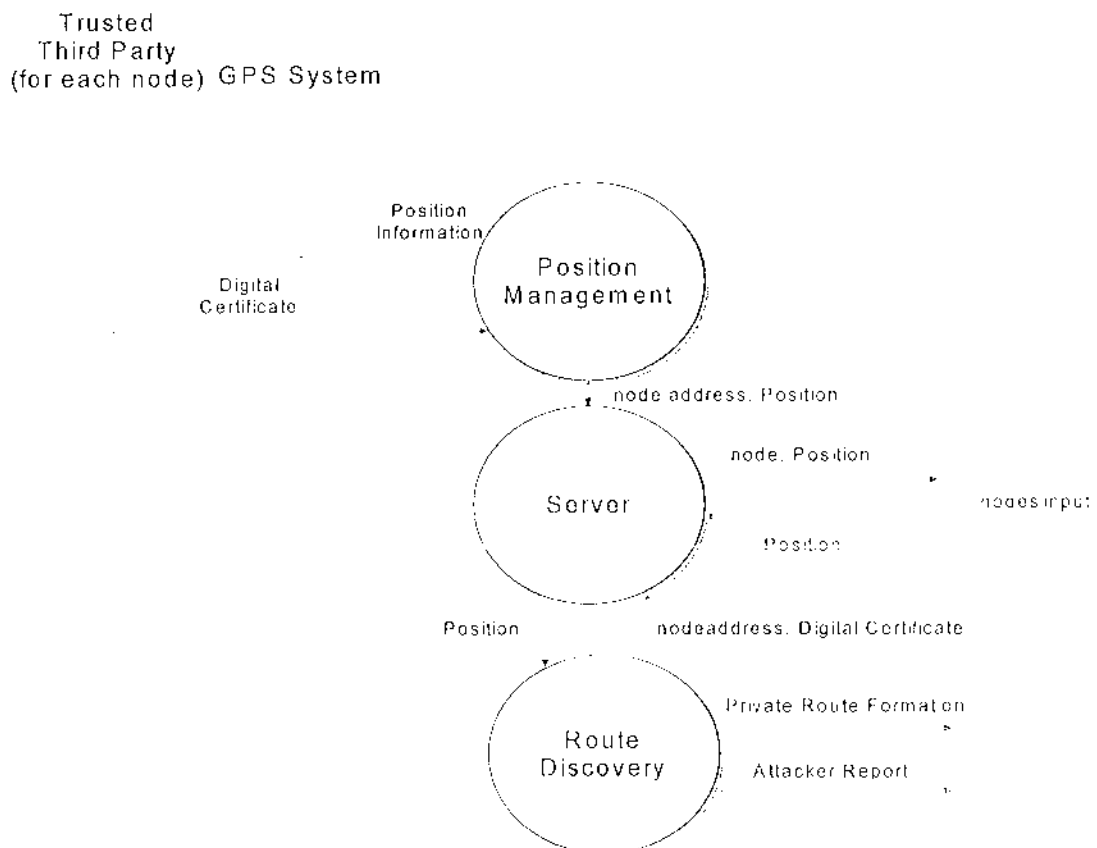
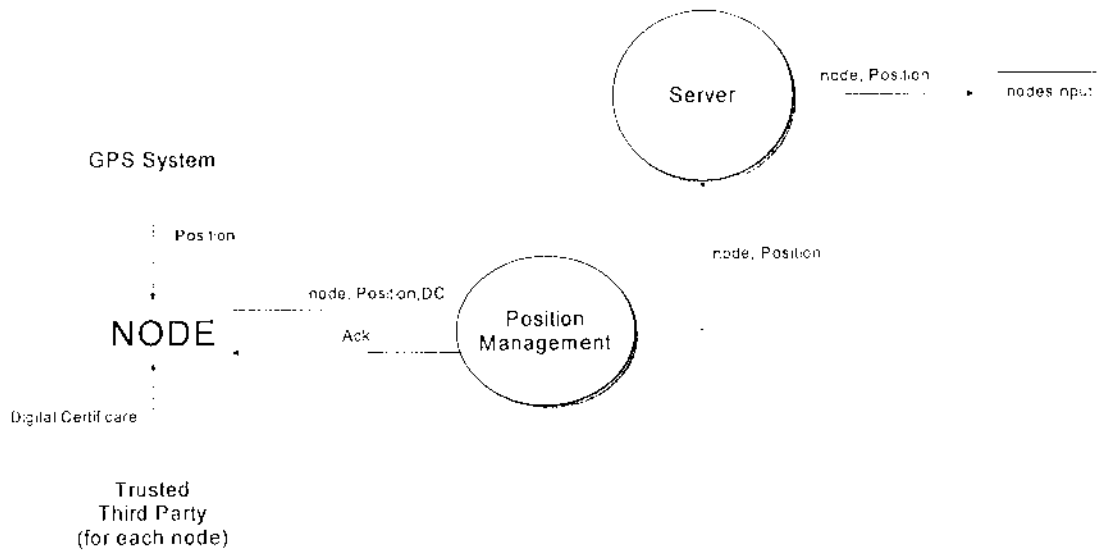
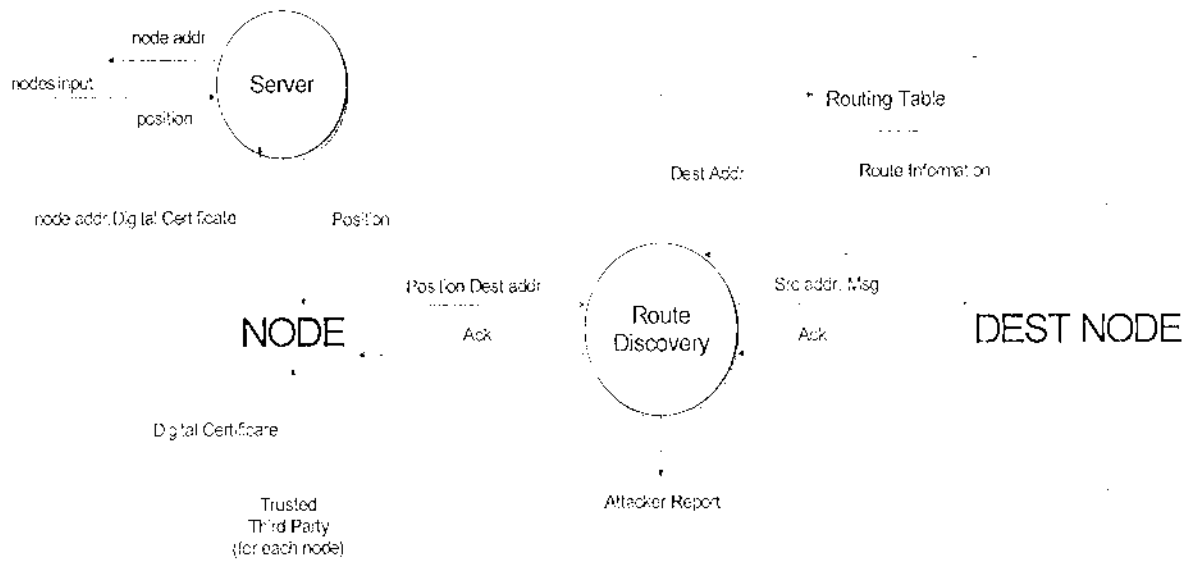


Fig 5.7 Level 1 DFD for AO2P Routing



**Fig 5.8 Level 2 DFD for Position Management**



**Fig 5.9 Level 2 DFD for Route Discovery**

---

---

*TESTING*

---

---

## 6. TESTING

Testing is done to detect the flaws in the software which helps in developing a quality product. This implies not only to the coding phase but to uncover errors introduced in all the previous phases.

### 6.1 PRODUCT TESTING:

To make GloMoSim work, copy the GloMoSim and parsec directories in to **C:\Glomosim** and add the following path in environmental variables.

**Path:**

**C:\Glomosim\glomosim\parsec\bin;**

Create a new environmental variable named PCC\_DIRECTORY.

**PCC\_DIRECTORY:**

**C:\Glomosim\glomosim\parsec**

and make sure that jdk1.2 or above is installed.

The following are the types of tests that were performed in our project.

**UNIT TESTING:** Each and every module is tested separately to check if its intended functionality is met. As our protocol is written using PARSEC functions, the command used to compile is as follows:

**>pcc module.pc**

This would generate **module.pi** files and **module.o** files.

Make sure to include header files at proper places.

**INTEGRATION TESTING:** It is the testing performed to detect errors on interconnection between the simulator GloMoSim and our protocol. This is done to ensure that it works in synchronization with each layer of simulator.



If the following code is not added in `C:\Glomosim\main\makefile.pc`,

.....

```
call pcc %parsecflags%-I...\include\-I...\network\-clock longlong -c
...\network\ao2p.pc
```

then the GloMoSim executable file will not be generated in bin directory, hence the GloMoSim will not work. The error observed will be: "No rule to make AO2P as target file". So adding of the specified code shown in the section 4.1 is necessary.

After making all the changes in the required files go to main directory and type

```
>makent.bat
```

now the `glomosim.exe` file will be created and it will be ready to work. If not given the changes will not be updated and hence no output is observed.

## 6.2 SYSTEM TESTING:

Using the simulator we generate the nodes and attacker nodes. Nodes send position request to the server to get the position of the destination. Then they send route request to identify the route. The attacker nodes are unauthorized. They try to access the packets in transmission. After implementing AO2P protocol it is tested that the packets through the attackers are discarded and a new route is identified to the destination. It is tested that the nodes can be run in any number of systems. We tested by placing the attacker in between two nodes, even though there is a delay in finding a new route the packets through the attackers is discarded.

---

*FUTURE  
ENHANCEMENTS*

---

## 7. FUTURE ENHANCEMENTS

At present we have implemented the AO2P protocol and presented in GloMoSim static Ad Hoc networks. In future, our protocol can be implemented and tested in Dynamic Ad Hoc networks. AO2P which does not rely on the exposure of the position information has been shown to provide a great amount of security.

We propose following two future research directions.

### **Privacy evaluation:**

Internal attackers are able to obtain pieces of question information of their targets. Based on this information they can estimate the trajectories of their target or reduces the anonymity set. Future work will include building analytical models for mobility and traffic based on which node anonymity can be quantified.

### **Security issues and mitigation techniques:**

In AO2P the next hop is determined by node contention. A malicious node can always use the most aggressive contention mechanism to become the next hop. Once it is included in a route, it can conduct different attacks, such as changing the position of destination in the routing request or dropping data packets after the route is build up. More seriously, protecting the privacy of intermediate nodes in AO2P makes it almost impossible to identify those attackers. A modified channel contention scheme will be developed. The next hop cannot be decided by a receiver itself, but by both the sender and the receiver. Information exchange is needed based on which a required trust between a previous hop and its potential next hop can be assessed.

---

---

## *CONCLUSION*

---

---

## 8. CONCLUSION

We have presented a new secured routing scheme based on reactive geographical routing protocol, called AO2P (Ad-Hoc On-Demand Position Based Private Routing Protocol), for ad hoc wireless networks to cope with the problems of overhearing and security attacks. The protocol deals with the quasi-stationary networks which have limited amount of movement. The proposed protocol utilizes the position information of each node to transmit the data and find a secured path.

The AO2P finds a path from source to the destination which can be trusted and free of attacks. The destination will update its new position to the source through the reverse route. The source can thus start a new routing discovery using the updated position information.

In AO2P the identities for the two ends of a communication are anonymous to other nodes. AO2P also protects the privacy for nodes acting as intermediate forwarders, as they do not need to expose any information during data delivery.

In general, AO2P combines both the advantages of reactive routing protocols which offer routing information with some latency since it usually needs to launch the route discovery process if it has no pre-discovered route thus reducing the bandwidth and geographical protocols which uses the location information of the nodes in the network to find the route, thus reducing control message overhead. Thus AO2P is better compared to other competitive routing protocols.

---

---

# *APPENDICES*

---

---

## 9. APPENDIX

### 9.1 SAMPLE SOURCE CODE:

#### HEADER FILES

```
#ifndef _AO2P_H_
#define _AO2P_H_
#include "ip.h"
#include "main.h"
#include "nwcommon.h"
#define BAD_LINK_LIFETIME      2 * RREP_WAIT_TIME
#define BCAST_ID_SAVE         30 * SECOND
#define REV_ROUTE_LIFE        RREP_WAIT_TIME
#define MY_ROUTE_TO           2 * ACTIVE_ROUTE_TO
#define RREQ_RETRIES           2
#define TTL_START              1
#define TTL_INCREMENT          2
#define TTL_THRESHOLD          7
#define AO2P_INFINITY          255
#define BROADCAST_JITTER      10 * MILLI_SECOND
/* Packet Types */
typedef unsigned char AO2P_PacketType;
#define AO2P_PREQ 0
#define AO2P_PREP 1
#define AO2P_RREQ 2
#define AO2P_RREP 3
#define AO2P_DATA 4
#define AO2P_ACK 5
#define AO2P_IREQ 6
#define AO2P_APREQ 7

typedef struct
```

```

{
    AO2P_PacketType pktType;
    NODE_ADDR servAddr;
    NODE_ADDR srcAddr;
        NODE_ADDR cdestAddr;
        int posi;
    int Certificate;
} AO2P_PREQ_Packet;

```

typedef struct

```

{
    AO2P_PacketType pktType;
    NODE_ADDR servAddr;
    NODE_ADDR destAddr;
        NODE_ADDR cdestAddr;
        int posi;

} AO2P_PREP_Packet;

```

typedef struct

```

{
    AO2P_PacketType pktType;
    int beastId;
    int posi;
        NODE_ADDR destAddr;
    //int destSeq;
    NODE_ADDR srcAddr;
    // int srcSeq;
    NODE_ADDR lastAddr;
    int hopCount;
} AO2P_RREQ_Packet;

```



```

typedef struct
{
    AO2P_PacketType pktType;
    NODE_ADDR servAddr;
    NODE_ADDR srcAddr;
    int Certificate;
} AO2P_APREQ_Packet;

```

```

typedef struct
{
    AO2P_PacketType pktType;
    NODE_ADDR srcAddr;
    NODE_ADDR destAddr;
    NODE_ADDR destAddrI;
    //int destSeq;
    int posi;
    int hopCount;
    clocktype lifetime;
} AO2P_RREP_Packet;

```

```

typedef struct
{
    AO2P_PacketType pktType;
    NODE_ADDR srcAddr;
    NODE_ADDR destAddr;
    NODE_ADDR CsrcAddr;
    NODE_ADDR CdestAddr;
} AO2P_ACK_Packet;

```

```

typedef struct

```

```

{
    AO2P_PacketType pktType;
    NODE_ADDR srcAddr;
    NODE_ADDR destAddr;
    NODE_ADDR CsrcAddr;
    NODE_ADDR CdestAddr;
    NODE_ADDR lastAddr;
    int bcastId;
    int posi;
} AO2P_IREQ_Packet;

```

typedef struct

```

{
    int numPostRequestSent;
    int numAttackerPostRequestSent;
    int numPostRequestReceived;
    int numAttackerPostRequestReceived;
    int numPostReplySent;
    int numPostReplyReceived;
    int numRequestSent;
    int numRequestReceived;
    int numIRequestSent;
    int numIRequestReceived;
    int numReplySent;
    int numReplyReceived;
    int numDataTxed;
    int numDataReceived;
    int numRoutes;
} AO2P_Stats;

```

```

void RoutingAo2pInit( GlomoNode *node, GlomoRoutingAo2p o2pPtr,
const GlomoNodeInput *nodeInput);
void RoutingAo2pFinalize(GlomoNode *node);
void RoutingAo2pHandleData(GlomoNode *node, Message *msg,
NODE_ADDR destAddr);
void RoutingAo2pHandlePostRequest(GlomoNode *node, Message
*msg);
void RoutingAo2pHandleAPostRequest(GlomoNode *node, Message
*msg);
void RoutingAo2pHandlePostReply(GlomoNode *node, Message *msg
,NODE_ADDR destAddr);
void RoutingAo2pHandleRequest(GlomoNode *node, Message *msg,
int ttl);
void RoutingAo2pHandleIRequest(GlomoNode *node, Message *msg,
int ttl);
void RoutingAo2pHandleReply(
    GlomoNode *node, Message *msg, NODE_ADDR srcAddr,
NODE_ADDR destAddr);
int Encode(int input,int key);
int Decode(int input);
int Encode(int* s);
int Decode(int* ss);
void RoutingAo2pRouterFunction(GlomoNode *node,Message *msg,
    NODE_ADDR destAddr,BOOL *packetWasRouted);
void RoutingAo2pPacketDropNotificationHandler(GlomoNode *node,
const Message* msg, const NODE_ADDR nextHopAddress);
void RoutingAo2pSetTimer(GlomoNode *node, long eventType,
NODE_ADDR destAddr, clocktype delay);
void RoutingAo2pInitiatePREQ(GlomoNode *node, NODE_ADDR
destAddr);
void RoutingAo2pInitiatePREP(GlomoNode *node, Message *msg);

```

```

void RoutingAo2pInitiateRREP1(GlomoNode *node, Message *msg);
void RoutingAo2pInitiateRREQ(GlomoNode *node, NODE_ADDR
destAddr, Message *msg);
void RoutingAo2pInitiateRREP(GlomoNode *node, Message *msg);
void RoutingAo2pInitiateRREPbyIN(GlomoNode *node, Message
*msg);
void RoutingAo2pRelayRREP(GlomoNode *node, Message *msg,
NODE_ADDR destAddr);
void RoutingAo2pInitiateDATA(GlomoNode *node, Message *msg,
NODE_ADDR destAddr);
void RoutingAo2pInitiateACK(GlomoNode *node, Message *msg);
void RoutingAo2pHandleAck(GlomoNode *node, Message *msg,
NODE_ADDR destAddr);

void RoutingAo2pInitiateIRREQ(GlomoNode *node, NODE_ADDR
destAddr, Message *msg);
void RoutingAo2pInitiateAPREQ(GlomoNode *node, NODE_ADDR
destAddr);
endif /* _AO2P_H_ */

```

### HANDLING PROTOCOL PACKET

```

void RoutingAo2pHandleProtocolPacket(GlomoNode *node, Message
*msg, NODE_ADDR srcAddr,
NODE_ADDR destAddr, int ttl)
{
    AO2P_PacketType *ao2pHeader =
(AO2P_PacketType*)GLOMO_MsgReturnPacket(msg);
    switch (*ao2pHeader)
    {
case AO2P_PREQ:

```

```

{
RoutingAo2pHandlePostRequest(node, msg);
break;
}
case AO2P_PREP:
{
RoutingAo2pHandlePostReply(node, msg, destAddr);
break;
}
case AO2P_RREQ:
{
RoutingAo2pHandleRequest(node, msg, ttl);
break;
}
case AO2P_RREP:
{
RoutingAo2pHandleReply(node, msg, srcAddr, destAddr);
break;
}
case AO2P_DATA:
{
RoutingAo2pHandleData(node, msg, destAddr);
break;
}
case AO2P_ACK:
{
RoutingAo2pHandleAck(node, msg, destAddr);
break;
}
case AO2P_IREQ:
{

```

```

RoutingAo2pHandleRequest(node, msg, ttl);
break;
}
case AO2P_APREQ:
{
RoutingAo2pHandleApostRequest(node, msg);
break;
}
default:
assert(FALSE); abort();
break;
}
} /* RoutingAo2pHandleProtocolPacket */

```

### INITIATE POSITION REQUEST FUNCTION

```

void RoutingAo2pInitiatePREQ(GlomoNode *node, NODE_ADDR
destAddr)
{
GlomoNetworkIp* ipLayer = (GlomoNetworkIp *) node->
networkData.networkVar;
GlomoRoutingAo2p* ao2p = (GlomoRoutingAo2p *) ipLayer->
RoutingProtocol;
Message *newMsg;
AO2P_PREQ_Packet *preqPkt;
char *pktPtr;
int pktSize = sizeof(AO2P_PREQ_Packet);
int ttl=1;
newMsg = GLOMO_MsgAlloc(node, GLOMO_MAC_LAYER, 0,
MSG_MAC_FromNetwork);
GLOMO_MsgPacketAlloc(node, newMsg, pktSize);

```

```

pktPtr = (char *) GLOMO_MsgReturnPacket(newMsg);
preqPkt = (AO2P_PREQ_Packet *) pktPtr;
preqPkt->pktType = AO2P_PREQ;
preqPkt->servAddr = destAddr;
preqPkt->srcAddr = node->nodeAddr;
preqPkt->cdestAddr = 7;
preqPkt->posi = 0;
preqPkt->Certificate = node->nodeAddr;
NetworkIpSendRawGlomoMessage(node, newMsg, ANY_DEST,
CONTROL, IPPROTO_AO2P, ttl);
ao2p->stats.numPostRequestSent++;
}

```

### INITIATE ATTACKER POSITION REQUEST

```

void RoutingAo2pInitiateAPREQ(GlomoNode *node, NODE_ADDR
destAddr)
{
GlomoNetworkIp* ipLayer = (GlomoNetworkIp *) node->
networkData.networkVar;
GlomoRoutingAo2p* ao2p = (GlomoRoutingAo2p *) ipLayer->
routingProtocol;
Message *newMsg;
AO2P_APREQ_Packet *apreqPkt;
char *pktPtr;
int pktSize = sizeof(AO2P_APREQ_Packet);
int ttl=1;
newMsg = GLOMO_MsgAlloc(node, GLOMO_MAC_LAYER, 0,
MSG_MAC_FromNetwork);
GLOMO_MsgPacketAlloc(node, newMsg, pktSize);
pktPtr = (char *) GLOMO_MsgReturnPacket(newMsg);

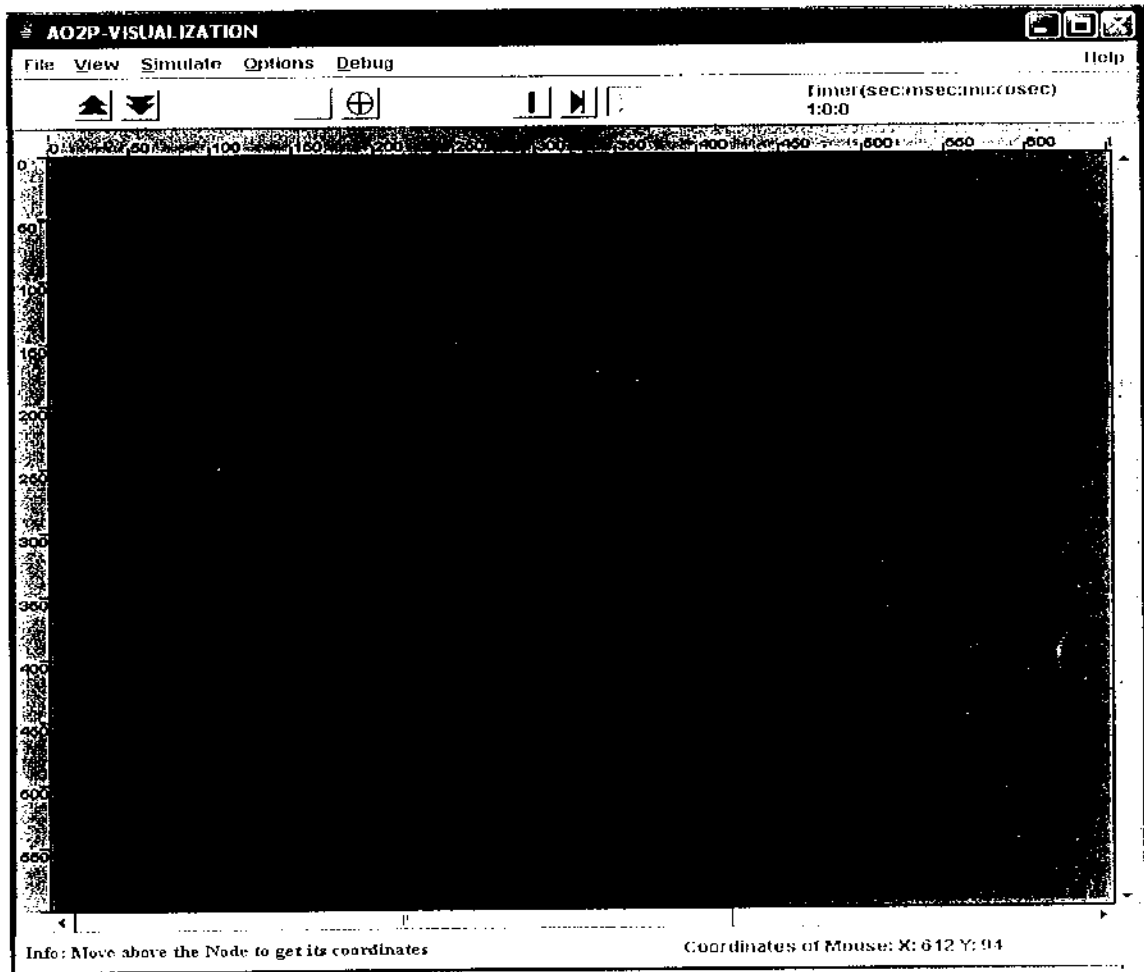
```

```
apreqPkt = (AO2P_APREQ_Packet *) pktPtr;
apreqPkt->pktType = AO2P_APREQ;
apreqPkt->servAddr = destAddr;
apreqPkt->srcAddr = node->nodeAddr;
apreqPkt->Certificate = node->nodeAddr;
NetworkIpSendRawGlomoMessage(node, newMsg, ANY_DEST,
CONTROL, IPPROTO_AO2P, ttl);
ao2p->stats.numAttackerPostRequestSent++;
}
```

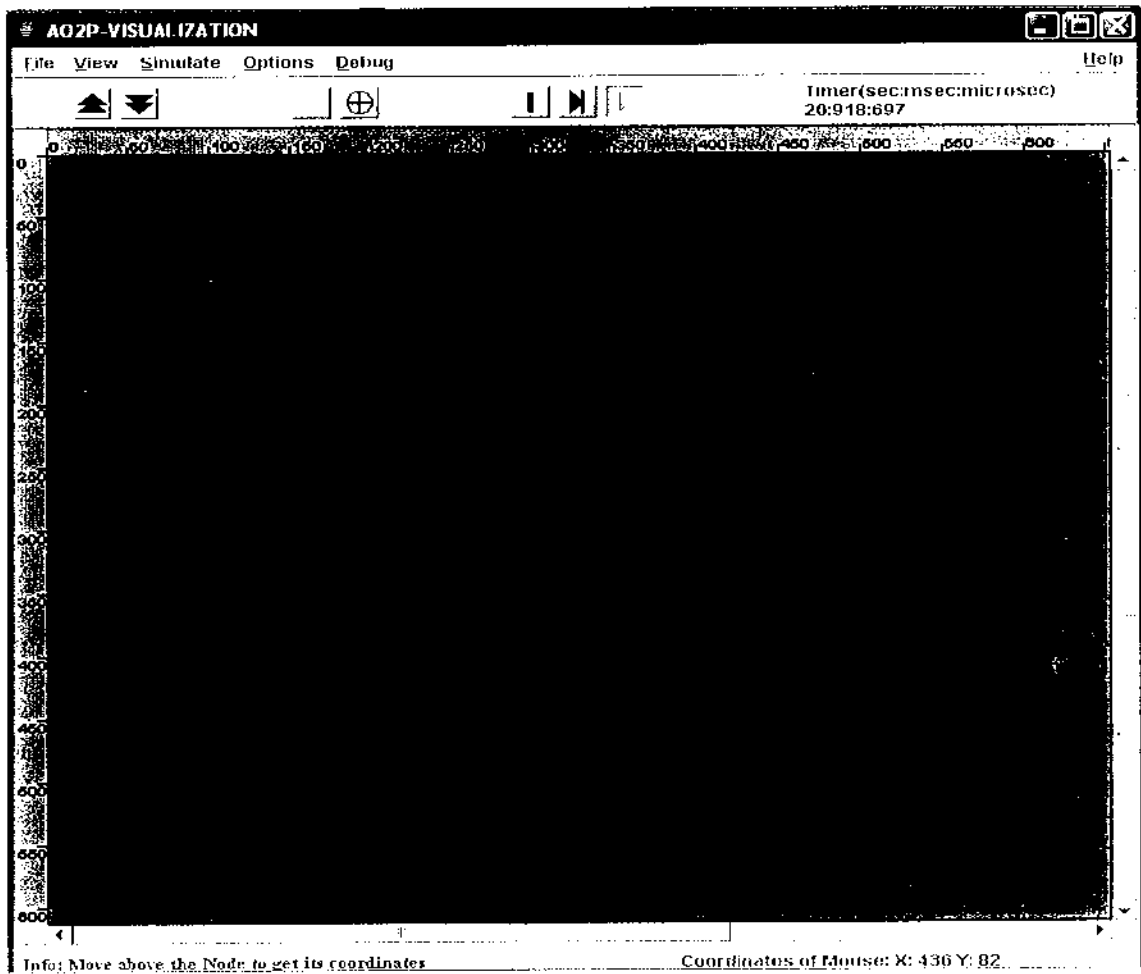


## 9.2 SAMPLE OUTPUTS:

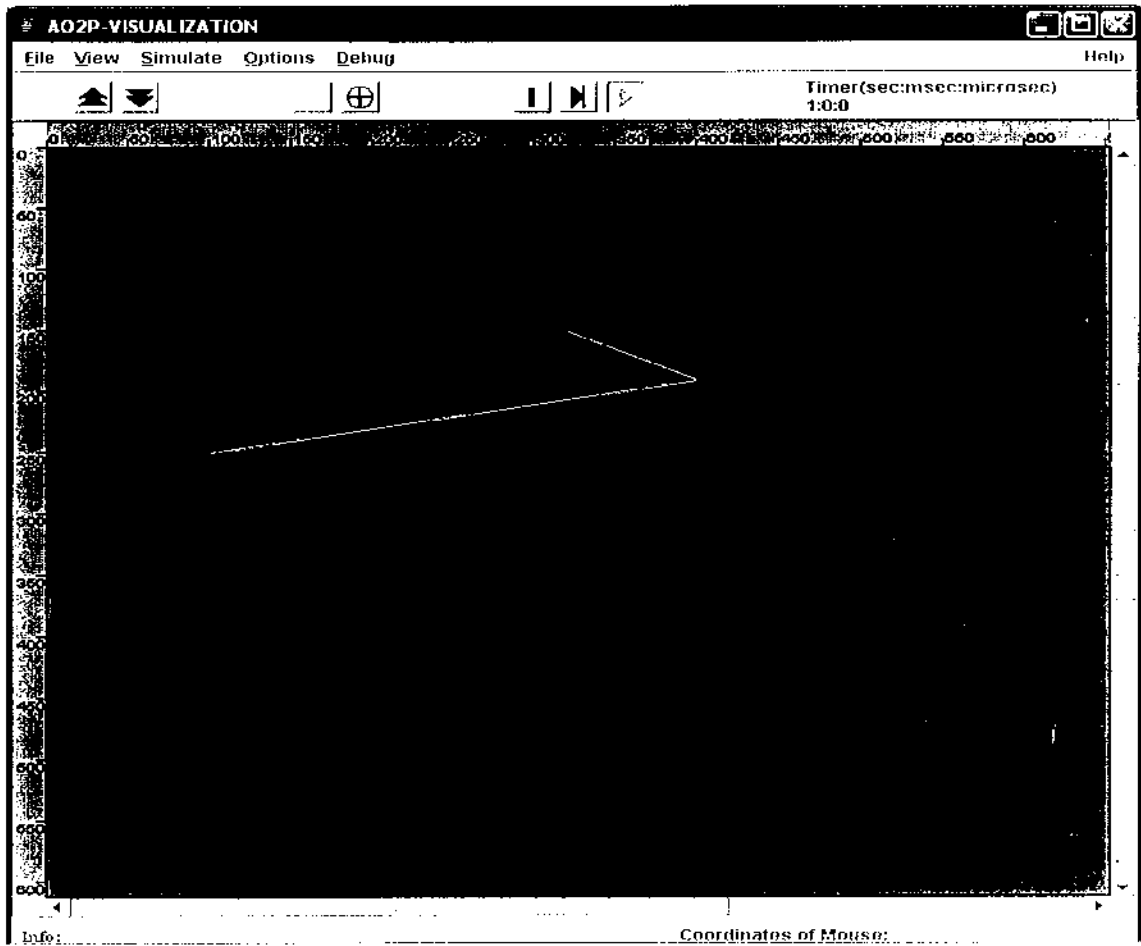
Sample output in GloMoSim Visualization tool showing the transmission range of each node:



Sample output in GloMoSim visualization tool showing the node connections:

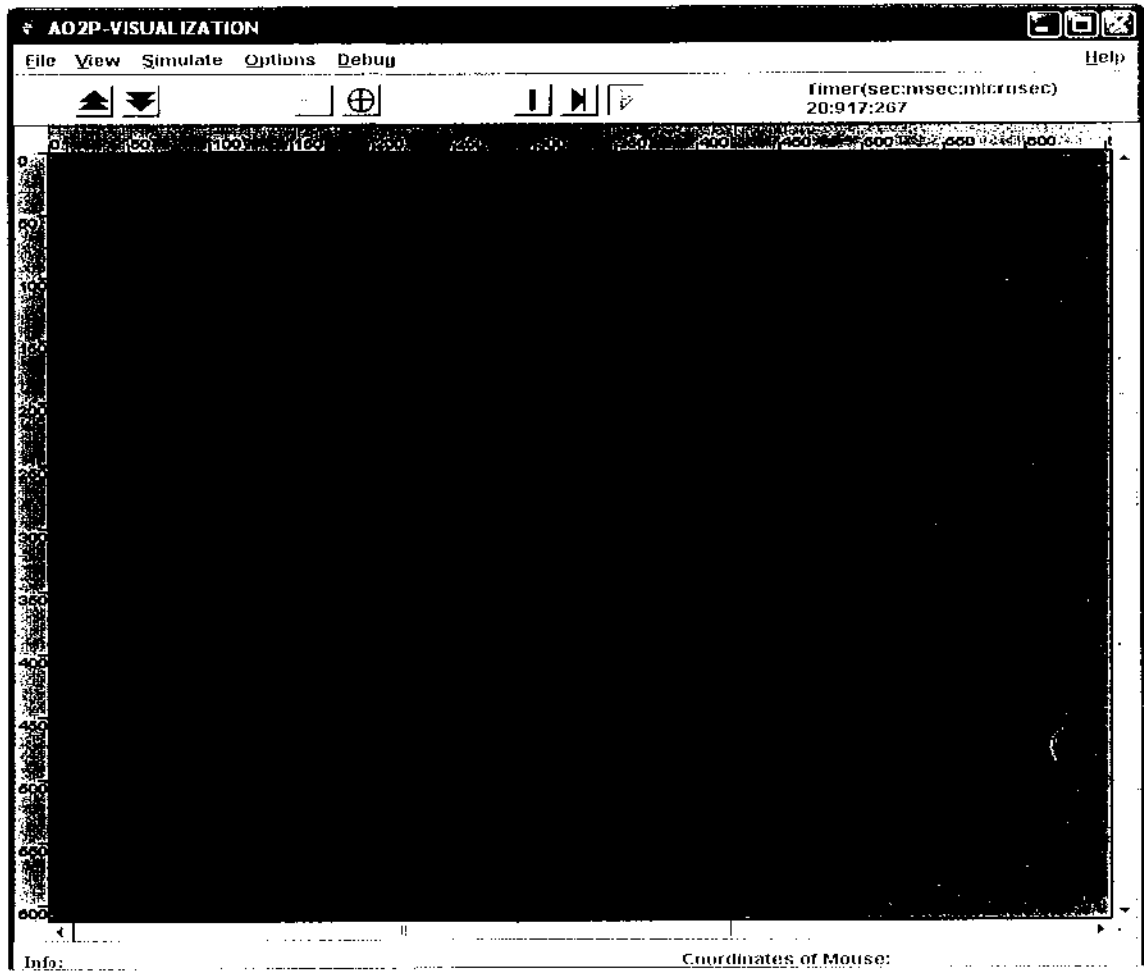


Sample output where External Attacker node-9 is broadcasting to nodes: 6 & 4

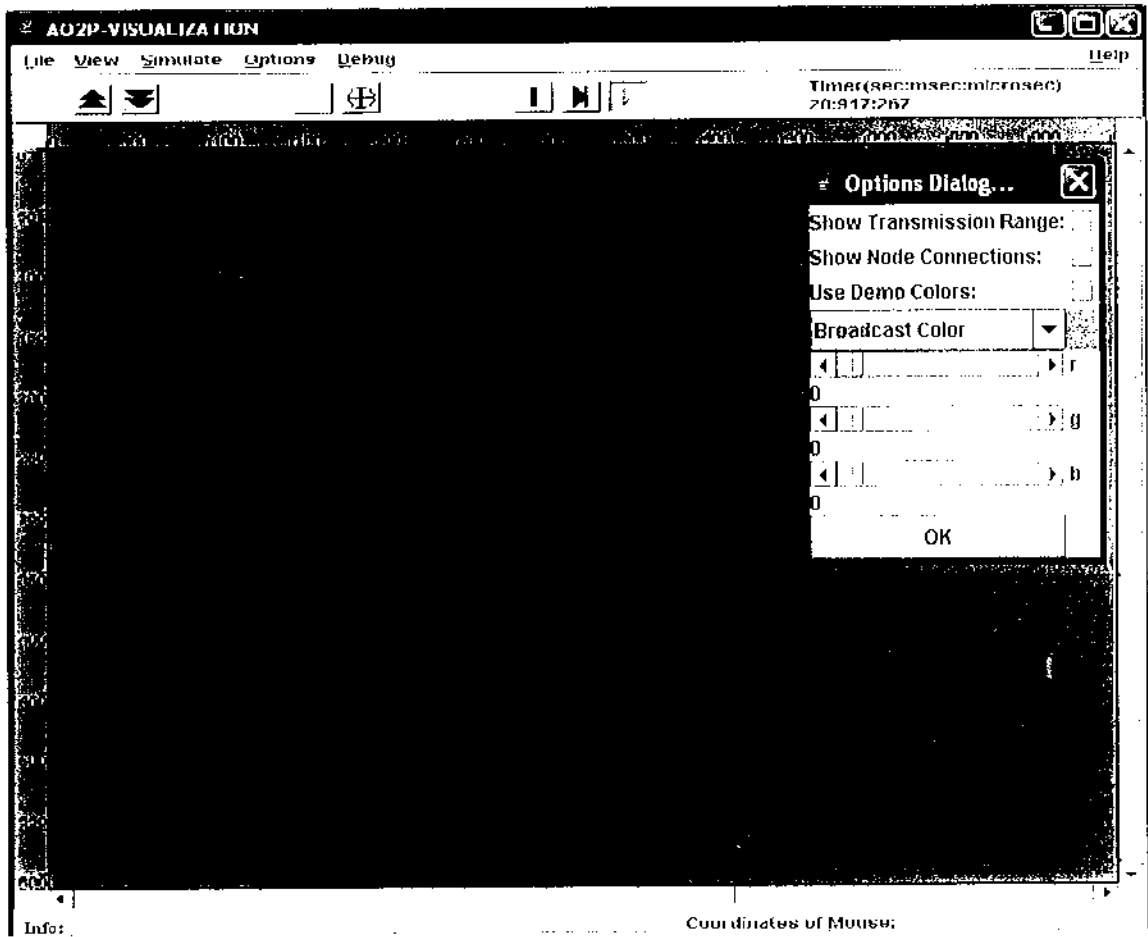


p 16/10

Sample output where source node-2 transmits messages to node-3:



Sample output showing options available in GloMoSim VT:



---

---

## *REFERENCES*

---

---

## 10. REFERENCES

1. Xiaoxin Wu and Bharat Bhargava, "AO2P: Ad Hoc On-Demand Position-Based Private Routing Protocol", *IEEE Transaction On Mobile Computing*, Vol 4, No 4, July/August 2005.
2. C.E. Perkins and L.M Royer, "Ad-Hoc On-Demand, Distance Vector Routing" *Proc. Second IEEE, Workshop Mobile Computing Systems and Applications*, 1999.
3. L. Xiao, Z.Xu, and X.Zhang, "Low-Cost and Reliable Mutual Anonymity Protocols in Peer-to-Peer Networks," *IEEE Trans. Parallel and Distributed Systems*, vol.14, no.9, pp.829-840, 2003.
4. M. Reed, P.Syverson, and D.Goldschlag, "Anonymous Connections and Onion Routing," *IEEE J. Selected Areas in Comm\**, special issue on copyright and privacy protection, vol.16, no.4, pp. 482-494, 1998.
5. P. Johansson, T. Larsson, N. Hedman, B. Mieleczarek. and M. Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks", *In Proceedings of the 5th International Conference on Mobile Computing and Networking, ACM MOBICOM '1999*, pp. 195-206
6. B. Karp and H.T.Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Network," *Proc. MOBICOM*, 2000.
7. Y.Xue, B.Li, and K.Nahrstedt, "A Scalable Location Management Scheme in Mobile Ad-Hoc Networks," *Proc. 26<sup>th</sup> Ann. IEEE Conf. Local Computer Networks*, 2001.
8. GloMoSim user manual,<http://pcl.cs.ucla.edu/projects/glomosim>