ARCHITECTURE AND PROTOCOLS FOR SERVICE AND APPLICATION DEPLOYMENT IN RESOURCE AWARE UBIQUITOUS ENVIRONMENTS

by

Anandha Gopalan

B. Sc. (Honors), St. Stephens' College, University of Delhi, 1996

M. Sc. St. Stephens' College, University of Delhi, 1998

M.S, University of Pittsburgh, 2002

Submitted to the Graduate Faculty of

Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2007

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF COMPUTER SCIENCE

This dissertation was presented

by

Anandha Gopalan

It was defended on

August 03, 2007

and approved by

ARCHITECTURE AND PROTOCOLS FOR SERVICE AND APPLICATION DEPLOYMENT IN RESOURCE AWARE UBIQUITOUS ENVIRONMENTS

Anandha Gopalan, PhD

University of Pittsburgh, 2007

Realizing the potential of pervasive computing will be predicated upon the availability of a flexible, mobility-aware infrastructure and the technologies to support seamless service management, provisioning and delivery. Despite the advances in routing and media access control technologies, little progress has been made towards large-scale deployment of services and applications in pervasive and ubiquitous environments. The lack of a fixed infrastructure, coupled with the time-varying characteristics of the underlying network topology make service delivery challenging. The goal of this research is to address the fundamental design issues of a service infrastructure for ubiquitous environments and provide a comprehensive solution which is robust, scalable, secure and takes into consideration node mobility and resource constraints.

We discuss the main functionalities of the proposed architecture, describe the algorithms for registration and discovery and present a power-aware location-driven message forwarding algorithm to enable node interaction in this architecture. We also provide security schemes to ensure user privacy in this architecture. The proposed architecture was evaluated through the use of simulations. The results show that the service architecture is scalable and robust, even when node mobility is high. The comparative analysis shows that our message forwarding algorithm consistently outperforms contemporary location-driven algorithms. Furthermore, this research work was implemented as a proof-of-concept implementation and tested on a real world scenario.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

Its been a long road to get here and along the way, I've been prodded, pushed and helped by the support and care provided by many a person. This thesis would never have been possible, if not for them.

I would first like to thank Dr. Taieb Znati, my adviser and guide, who has stood by me through this effort, without whose support and constant guidance, it would not have been possible to see the finish line. Even though he was very busy, he still found the time to help, guide and motivate me when I needed it.

I would like to thank Dr. Rami Melhem, Dr. Kirk Pruhs and Dr. Prashant Krishnamurthy for agreeing to be on my thesis committee and for their helpful support in the shaping of this document.

My office mates (Hui, Guanfeng, Chatree, Hammad and Octavio) must also be thanked for their help. They were ever ready to provide a helping hand, discuss ideas and topics that would prove helpful to me. They are also to be thanked for spending countless hours as my audience to help and refine my presentation skills. It was a pleasure working with them.

Our Technical Staff (Bob Hoffman, Russ Howard, Terry Wood and Chris Mason) should be thanked for catering to my idiosyncratic whims and fancies. They were very patient and extremely helpful in solving the numerous technical issues. It was also a pleasure working with them as part of the tech team in the department.

I would like to thank Loretta, Kathy, Nancy, Kathleen and Keena for helping me resolve multiple issues with the department and for patiently answering my many questions about procedures in the department, helping me with day to day issues (if I had any) and of course for helping me with boxes to aid in my shifting. I wish Lorretta the very best in her retirement.

My friends Piyush, Joyee, Manu, Rajesh, Chako, Swagata, Barun, Ummi and Vijay have stood by me and provided me constant guidance and never ending support, I am immensely grateful for this and more. I consider myself to be immensely lucky to have such wonderful friends around.

Finally, I would like to thank my parents, aunt, uncle and family for their immense patience and never ending support. I would especially like to thank my brother Hemmant, for giving me enough and more of his time whether it was to discuss ideas, correct my work or to generally chat. Many a time, just having them in my thoughts helped. It wouldn't have been possible without them.

1.0 INTRODUCTION

The vision of ubiquitous computing was first articulated in 1991 by Mark Weiser, then chief technology officer for Xerox's Palo Alto Research Center. In his paper, he stated that "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it" [40]. Significant advances in engineering and communication technologies over the past 16 years have made Weiser's vision a viable one. There are several ubiquitous computing projects in industry (AT&T, Intel, HP and IBM), as well as academia (Project Oxygen at UC Berkeley [51], Endeavour at MIT [15], Project Aura at CMU [50] and Portolano at the University of Washington [49]) that go a long way towards making pervasive and ubiquitous computing a reality.

Technological advances in engineering and communication have paved the way for a new generation of embedded wireless devices. These devices range from small inexpensive lightweight sensors, with limited memory and computational capabilities, to resource-rich devices which can support significantly enhanced functionalities. A number of these devices can be deployed on a large scale for sensing and in-situ processing of spatially and temporally dense data, and for carrying out specialized functions. The objective of such infrastructureless networks is to support increased mobility, flexibility, and lower cost of managing the resources in comparison to infrastructured networks. These networks are unique, in the sense that a participating device can function both as a host as well as a router, thereby dynamically creating paths between network devices. Unlike a fixed wireless network, however, locating a device becomes difficult as users may exhibit high levels of mobility.

These infrastructureless heterogeneous networks of embedded systems have great potential for significant impact on a wide range of time-critical applications, such as pervasive health care, infrastructure protection, homeland security, and real-time environment monitoring. Demand for continuous real-time monitoring for critical infrastructure protection, border control, and disaster management, among many other time-critical applications, has become increasingly important. Continuous monitoring and control of key infrastructure, such as power grid, transportation networks, water supply, oil and natural gas pipelines, and major railroads, is crucial for protection against threats coming from natural causes, by-products of human activities or from deliberate actions undertaken for criminal purposes. Traditional monitoring technologies require considerable human intervention and are usually inadequate to protect against unexpected failures and adversarial attacks, such as denial of service. Rapidly deployable, self-configuring networks of wireless devices have the potential for providing the information needed to assist rescue operations, by locating survivors, identifying affected areas, and organizing the collaborative efforts of the response team members.

There is an increase in the deployment of wireless devices for purposes of ubiquitous and pervasive health care. This technology will simplify monitoring and treatment of patients by providing the ability to monitor patients over long periods of time without the intervention of medical or health-care professionals. Such up-to-date information on a patient's condition will enable both patients and medical professionals to better understand how to manage their activity to avoid or anticipate problems and provide preemptive care.

There are typically three modes of data delivery: polling, continuous and event-driven. In the polling mode, data is communicated only after a request is issued by a node indicating its interest to receive the latest value of a particular attribute. In the continuous mode, data is sent to devices in a periodic and continuous manner. The time interval for sending the data periodically is typically application-dependent and negotiated between the sender and the receiver. Finally, in the event-driven mode, communication between devices occurs only in response to an event. These modes of data delivery are not necessarily exclusive of each other, as a combination of the three modes may be required by the application.

The use of any one of the modes of data delivery or several of them depends on the demands of the application. A doctor monitoring the heartbeat of a patient relies on polling the device once in a while. If the doctor detects an anomaly and would like to get more readings to preempt any potential problems he/she increases the rate of sending the request for the required data. The file sharing application scenario also uses polling based data delivery, since nodes search for and retrieve files based on requests. An application to maintain room temperature in the ICU (Intensive Care Unit) would use the continuous data delivery mode to continuously monitor the state of the system. Such an application would also make use of an event driven delivery mechanism when an unusual event happens, such as the case when the temperature increases higher than a pre-determined threshold. This event triggers the data delivery mechanism to alert the respective emergency response teams. These three data delivery scenarios along with their interfaces and functions are depicted in Table 1.1. These interfaces and functions provide the mechanism by which data delivery can be controlled and changed to suit the application's needs.

Table 1.1: Data Delivery Scenario Interfaces

Name	Functionality
continuous_send (ti)	Continuously send data to the respective devices during each time interval (*ti*)
change_interval (newti)	Set the new time interval for continuous_send () to *newti*
poll_send (request)	Send a *request* for the data
poll_respond (request)	Respond to the *request* by sending the required data
event_send (event)	Respond to the *event* by sending the required information

The application scenarios mentioned above would require a mechanism for locating, requesting and downloading information. For this purpose, a service architecture is needed. A service architecture is a collection of data structures, protocols and mechanisms that allows heterogeneous devices in an ubiquitous environment to discover each other and discover the resources and services available in the network. This also allows applications and devices to register and discover the interfaces that allow them to choose the the best data delivery scenario which suits their respective needs. This is achieved through the process of service registration and discovery respectively. Service registration allows network entities to advertise their presence when they enter the network. These advertisements include additional contact information, and also the description and attributes of the entity. Service discovery allows the nodes in the network to discover the registered services.

Although typical service discovery frameworks work at a higher layer than routing, it is imperative to integrate service registration and discovery with traffic forwarding. Unlike traditional wireline networks, node mobility is a factor in wireless networks and has to be accounted for while designing the protocols for service registration and discovery. The goal of this research is to address the fundamental design issues of a service infrastructure for ubiquitous environments and provide a comprehensive solution which takes into consideration node mobility and resource constraints.

1.1 THESIS STATEMENT

The concepts of Most Likely Residence, Virtual Registry and Location Vector will enable the development of a framework, a set of protocols and appropriate mechanisms that allow for scalable, robust, resilient and secure service deployment in ubiquitous environments.

1.1.1 Validation

This work is validated through the design and evaluation of a service architecture for large-scale service and application deployment in ubiquitous environments. More precisely:

1. It designs and develops an efficient, robust, scalable and secure framework for large-scale service and application deployment in ubiquitous environments. This architecture is also evaluated through the use of simulations.
    - In order to understand and evaluate the performance of the service architecture, we perform a sensitivity analysis of the proposed architecture. Extensive simulations for a variety of node densities and node mobilities were performed to study the behavior of the service architecture under different network settings.
2. It designs and develops a power, resource, and location aware traffic forwarding algorithm for the service architecture. This algorithm uses a priority based scheme that imposes a priority on the neighboring nodes such that nodes which are more in line with the direction of the destination have higher probability to forward the message. This priority

is also closely tied to the residual energy-level of the intermediary node to increase network lifetime. The algorithm is evaluated through the use of simulations.

- In order to evaluate the performance of the message forwarding algorithm, its performance will be compared to GPSR, LAR and AODV. Simulations were performed to measure the throughput achieved for each protocol for a network consisting of mobile nodes while varying different network parameters.

3. It designs a security scheme to ensure user privacy in the proposed service architecture. Three schemes are developed, each of which serve a different purpose and the best scheme can be chosen depending on the network conditions.
4. It provides a proof of concept implementation of the proposed architecture that shows its ability to perform in a real world scenario.

   - Architectures and protocols developed for ubiquitous environments are normally tested using simulations. Without actual implementation, it is difficult to perceive how efficient and effective the protocol would be in the real world. For this reason, the proposed service architecture along with the message forwarding protocol were implemented on Linux and tested under real world conditions.

There are several challenges that arise when trying to develop a framework for service and application deployment in ubiquitous environments. These challenges are related to the development of several capabilities, such as: object registration, object discovery, mobile node location, and traffic routing and forwarding. The requirements and issues associated with these capabilities are discussed below.

1.1.2 Object Registration and Discovery

Object registration is the mechanism by which a node, managing a collection of related objects, registers these objects with the network. Typically, the object information along with the node information are registered with some specific node (or nodes) in the network.

Object discovery is the mechanism by which nodes discover the objects of interest that are available in the network. Typically, nodes attempt to locate the node with whom the object is registered with and queries for the object of interest. If a match occurs, the corresponding information on locating a node that owns the object is returned.

Despite advances in areas of routing and media access technologies for pervasive and ubiquitous computing, little progress has been made toward large-scale deployment of services and applications in such environments. The lack of a fixed infrastructure, coupled with the time-varying characteristics of the underlying network topology, makes service delivery challenging.

Although typical service discovery frameworks work at a higher layer than routing, it is imperative to integrate registration and discovery services with traffic forwarding. Unlike traditional wireline networks, node mobility is a factor in wireless networks and has to be accounted for while designing the protocols for object registration and discovery.

The existing service discovery protocols like the Service Location Protocol [14] and the Simple Service Discovery Protocol [75] are designed for and work well in LANs, but are not suitable for ubiquitous environments due to their reliance on an existing network structure. Distributed Hash Table (DHT) based protocols such as, CAN [52], GHT [62], GLS [30] and EKTA [23] provide distributed data indexing algorithms for ad-hoc networks. The limitation of these research works is the fact that they do not consider node mobility. Also, in the case of CAN and GLS, the DHT is based on virtual co-ordinates and does not reflect the underlying physical topology of the network.

A new service architecture must be developed that allows the network to support the basic functionalities necessary to enable a computational platform for node interaction without compromising on the convenience offered by infrastructureless networks.

1.1.3 Traffic Forwarding

Traffic routing and forwarding is the mechanism by which application and control traffic reaches the intended destination. Due to the lack of an existing infrastructure, traffic forwarding in an infrastructureless network is of high priority. In an infrastructureless network, forwarding traffic between the source and the destination would require the use of intermediary nodes. There are several routing protocols for infrastructureless networks that have been suggested by researchers [34, 46, 47, 41, 25] and these can be broadly categorized into three main categories: pro-active routing protocols, re-active routing protocols and hybrid routing protocols.

There is however, a new class of routing protocols that use the position of a node in space rather than the topology of the network. These routing protocols are classified as location-based routing protocols. Location-based routing protocols take advantage of the fact that nodes know their location (using a service similar to GPS [13]) and use this information to optimize the routing by sending messages in the direction of the destination rather than broadcasting it. Examples of location driven routing protocols are: LAR [35], GPSR [6] and DREAM [56]. LAR and DREAM cannot be used for ubiquitous environments since they flood the network with location updates and hence are not scalable.

The strategy used to forward traffic efficiently in the service architecture must take into consideration the time-varying dynamics of the network, node mobility and power-consumption. The trade-offs between these important design factors and network characteristics must be recognized and alternatives carefully evaluated.

1.1.4 Node Mobility

An ubiquitous computing environment consists of a collection of collaborative nodes. One of the main defining characteristics of these nodes is their mobility. Unlike nodes in the traditional LANs, where each computer/laptop is associated with a network port, nodes in an infrastructureless network are free to move about. Nodes in the network can be static, move once in a while or be continuously on the move.

Some schemes that have been developed to handle node mobility include [27, 26, 57]. These use the concept of home agents (or) home regions. Each node in the network is mapped to an area (using a hash function) in the network that is designated as its home agent (or) home region. The home region holds the location information about the mobile nodes which map to this location. These schemes are expensive in terms of communication costs, since a mobile node constantly updates its location information by sending updates to its home region.

Node mobility, coupled with the limitation of computational and communication resources, bring about a new set of challenges that need to be addressed in order to enable an efficient, robust and scalable architecture for service and application deployment in ubiquitous environments. The mobility of nodes in the network requires information to be stored in the network that clearly goes beyond the typical information stored for a service architecture. In addition to the service information, the mobility information of a node must also be stored in order to facilitate node interaction. This mobility information however, changes dynamically, as the node moves from one location to another. Efficient mechanisms must, therefore be in place to update this information as nodes move. Thus, it is imperative to integrate mobility management with the traffic forwarding protocol and the protocols for service registration and discovery.

1.1.5 Security

In an ubiquitous computing environment, users often interact directly with the environment through portable devices that they carry. As the users move around, they can still keep in touch and interact with the ubiquitous environment through the use of their portable devices. The information exchanged between the user and the environment often consists of user identity and location information. It is imperative to protect this information to guarantee user privacy. The lack of user privacy may deter users from using the ubiquitous computing environment.

Most of the research work done on security for pervasive and ubiquitous environments has been in the area of secure routing, data integrity and key management. These include: shared-key authentication [77, 22, 8], secure routing [16] and multi-path key establishment schemes [21, 22]. These schemes are computationally expensive for our purpose of protecting user privacy. To ensure user privacy in an ubiquitous computing environment, it is enough to ensure that the location information of the user is protected. This can be achieved by a "shadowing" mechanism whereby a user can be perceived to have multiple locations. The schemes developed to ensure user privacy must also take into account node mobility and the resource constrained environment.

1.2 THESIS CONTRIBUTION

The objective of this thesis is to build a robust, scalable, efficient and secure framework for service and application deployment in ubiquitous environments. We present the system model in Section 1.2.1 and the guiding approach in Section 1.2.2, before listing the thesis contributions in Sections 1.2.3 - 1.2.5.

1.2.1 System Model

In this section, we list the assumptions that the system model uses for all the algorithms and protocols developed. The network service area spans a geographical area that consists of a mix of both fixed, as well as mobile nodes. Some of the nodes in the network are assumed to own objects (documents, music, patient reports, maps). Each entity (node or object) in the network is uniquely identifiable by its id. Every node in the system knows its own location (using global or local co-ordinates) and also the boundaries of the network service area. The uniform hash function used to register and discover objects is known to all the nodes in the network. Each node in the network has a public/private key pair and it is not possible to guess the identity of the node from its public-key. Routing in the network is location based, where the message is sent to an (x,y) location rather than a particular node.

1.2.2 Guiding Approach

The basic approach is to use "wired" services as and when available and augment this by using "mobile" services when necessary. In an ubiquitous computing environment, it is often the case that there are some tethered services available (e.g: patient monitoring station, fixed sensors, desktops etc.). Availability of tethered services is an advantage to the infrastructureless network since these services have long lives and have fixed points of entry into the network. Rather than treating the mobility of the nodes in the network as a hindrance, we use mobility as an advantage. The advantage of giving preference to tethered services is the longevity of the service, while using "mobile" services in the absence of fixed services provides us with a solution which is efficient and has predictable results.

This thesis addresses the need for new service registration and discovery models and proposes a mobility-aware service architecture that is well-suited for scalable, robust, efficient and secure service deployment in pervasive and ubiquitous environments. The logical organization of the proposed service architecture is shown in Figure 1.1.

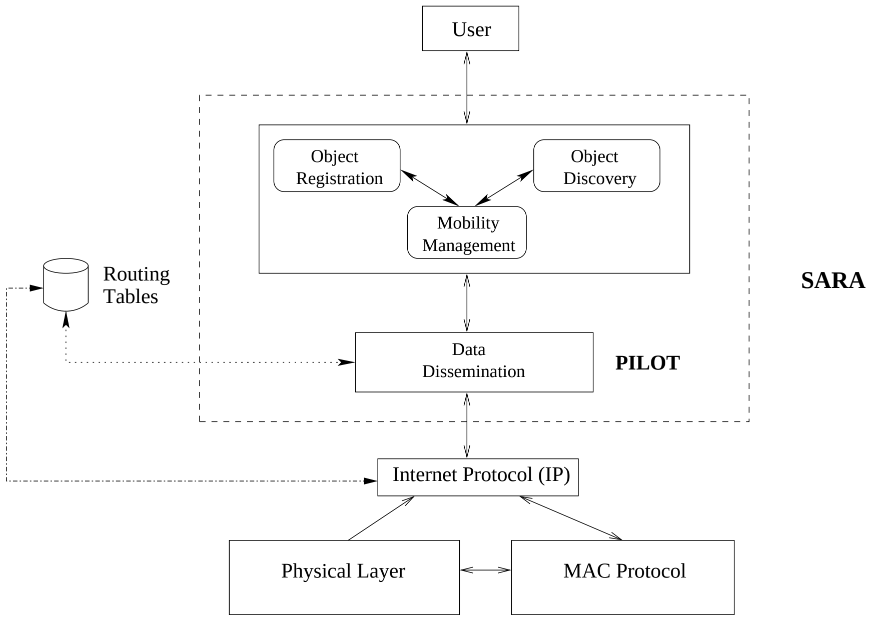


Figure 1.1: Overview of the Design

The highest layer in the framework is the user who uses the exported primitives of object registration and discovery to either register his/her objects with the network or discover objects from the network. Due to the mobility of nodes in the network, the object registration and discovery components make use of the mobility management component to locate the mobile node. These three components make use of the data dissemination (PILOT) layer to be able to route application or control traffic to the intended destination. Once a packet is available to be forwarded, the data dissemination layer passes the packet to the Internet Protocol (IP) layer. This layer is responsible for creating IP packets that are directly sent to the MAC layer, which in turn injects this packet into the network. At the lowest level in the framework are the physical and medium access control (MAC) layers that interact with each other to send/receive packets. Sections 1.2.3 - 1.2.5 now detail the contribution of this thesis.

1.2.3 SARA: A Service Architecture for Resource Aware Ubiquitous Environments

The first contribution of this thesis is *SARA*, a service-architecture for ubiquitous environments that does not assume a fixed infrastructure and imposes no location restrictions on the nodes and objects in the network.

The basic tenet of this architecture revolves around the concepts of *virtual registries*, *most likely residence* and *location vector*. A virtual registry is a dynamically created administrative domain that enables object registration and discovery. The extent of a virtual registry is such that it encompasses at least $\mathcal{K}$ (threshold) nodes. The information in a virtual registry is maintained by its member nodes. The *most likely residence* of a node is the physical area where the node is likely to be located most of the time. For example, the most likely residence of a fixed node is its physical location. This is registered by the node with the network and is used as a *congregation* point by nodes to contact other nodes. In the case of a mobile node, the node also registers its *location vector*. The location vector of a mobile node is a dynamic time-dependent vector that represents the most likely physical location of the node at a given time, thus reflecting user activity. The *primary* advantage of this approach is that each node can choose to provide its *own* mobility prediction model, which it deems to be most appropriate to its current activity, rather than using a network-wide model which may not be applicable to specific itineraries and situations.

Object registration and discovery are achieved by hashing the *object id* to obtain the physical co-ordinates of a *point* (P) within the network service area. The set of mobile nodes in the virtual registry containing P assume the responsibility of maintaining information about the object. The basic *design principle* for our scheme is to use *geographical* mapping for the hashing as opposed to *node* mapping since nodes are mobile. While bootstrapping, a node only needs to know the hash function that is used to register and locate objects in the network. To ensure that there are no *hot spot*s in the network due to hashing, the hash function is chosen to be a *uniform* hash function [12].

1.2.4 PILOT: A Power-Aware Location Driven Traffic Forwarding Algorithm

The second contribution of this thesis is *PILOT*, a new data dissemination and propagation algorithm for the proposed service architecture that forwards traffic in a power-aware location-directed manner.

To limit flooding in the network, PILOT uses the knowledge about the location of the source and the direction of the destination to forward traffic in a truncated cone-shaped manner towards the destination. The intermediary node to forward traffic is chosen by using a priority-based scheme that imposes a priority on the neighboring nodes in such a way that nodes which are more in line with the direction of the destination have higher probability to forward the message. This reduces the delay that traffic suffers on its way towards the destination. This priority is also closely tied to the residual energy-level of the intermediary node to maximize network lifetime. Consider the case of two nodes, similar with respect to their position from the source and the destination; the node with higher energy will have a higher probability to forward the message towards the destination.

1.2.5 Security

The third contribution of this thesis are schemes to ensure user privacy in the proposed service architecture. These schemes are light weight and were developed while taking node mobility and the resource constrained environment into consideration. The proposed security mechanisms are divided into three schemes: *Multiple Location Vector* scheme, *Node-Proxy Based* scheme and *Random-Proxy Based* scheme. The Random-Proxy Based scheme works on the assumption that each node in the network has a public-key/private-key pair.

In the *Multiple Location Vector* scheme, the node registers multiple *location vectors* with the *virtual registry.* This allows the mobile node to "mask" its current location by using multiple different locations in the network. When using the *Node-Proxy Based* scheme, a mobile node registers the *location vector* of its proxies. These proxies are chosen by the node during the bootstrap process. A node's location vector is not associated to a particular node since the id of a node is associated with the location vector of a proxy node. In the *Random-Proxy Based* scheme, the *virtual registry* replying to a query for a node's location vector constructs a “path” of nodes to traverse before reaching the destination node. In this scheme, the location information of each node along the path is encrypted by using the public-key of the preceding node.

Depending on the network conditions and the necessary constraints, anyone of the above security schemes may be used to ensure user privacy in the service architecture.

1.3 THESIS ORGANIZATION

The rest of the thesis is organized as follows: Chapter 2 details the research related to this thesis in the areas of: Medium Access Control (MAC), Distributed Hash Tables (DHT)s, routing and service discovery in infrastructureless networks. In Chapter 3 we present SARA, a resource and location aware framework to support service and application deployment in ubiquitous environment. We present the components of SARA and detail the building blocks of SARA (Section 3.2) along with the services used for registry creation and management, object registration and discovery, mobile node location and node interaction (Section 3.3). Chapter 4 details the power aware message forwarding algorithm used in SARA. Chapter 5 details the schemes used to ensure user privacy in SARA. Chapter 6 contains the implementation details of the proof-of-concept implementation of SARA with PILOT as its underlying message forwarding protocol. Chapter 7 details the simulation environment and provides a sensitivity analysis of SARA and a comparative analysis of PILOT. Chapter 8 concludes this thesis by outlining the contributions of this thesis and providing directions for future work.

2.0 BACKGROUND AND RELATED WORK

This purpose of this chapter is to present a detailed review of the literature and the background that are related to this thesis. The field of pervasive and ubiquitous computing is relatively new, yet the available literature has grown appreciably in the recent past. As a growing number of applications of ubiquitous computing become apparent, the available literature continues to grow rapidly. The growth of ubiquitous and pervasive computing has been rapid, aided mainly by the advances in hardware systems, availability of unlicensed radio spectrum, high cost and limitations of infrastructured wireless networks, advances in routing, advances in MAC (Medium Access Control) layer technology, and advances in security for infrastructureless networks.

The background and literature directly related to this thesis has been split into different sections. Section 2.1 details the background and advances in Medium Access Control technology. The available body of literature ranges from the traditional CSMA/CA MAC protocols to energy efficient MAC protocols such as, SMAC [73]. Section 2.2 details the research work related to Distributed Hash Table (DHT)-based systems. In this section, we provide a brief overview of DHTs and provide the examples of CAN, GLS and GHT. Section 2.3 describes the research work related to routing protocols for infrastructureless networks. In this section, we break up the body of routing protocols into different categories, namely: pro-active, re-active, hybrid and location-based. Examples are provided for each of these different kinds of routing protocols. Section 2.4 concludes this chapter by providing in detail the research advances related to service discovery in infrastructureless networks.

2.1 MEDIUM ACCESS CONTROL (MAC)

The Medium Access Control (MAC) layer provides the functionality using which access to the shared medium is granted in a fair and efficient manner. The earliest wireless networks were packet radio networks that used protocols such as ALOHA to grant access to the wireless channel [5]. ALOHA was not effective in allocating access to the wireless channel, since the probability of collision was high. Other techniques included time division multiple access and frequency multiplexing.

An improvement on the above schemes was the carrier sense multiple access scheme, in which nodes listen to the channel and transmit only if the channel is free. The nodes that wish to participate in a wireless communication negotiate a *RTS* (request-to-send)-*CTS* (clear-to-send) handshake before transmission. This handshake is to ensure that the wireless channel is free and is necessary since collision detection is very difficult in wireless networks. This protocol is called: carrier sense multiple access with collision avoidance (CSMA/CA) and is used in the IEEE *802.11* standard for wireless networks [28].

A major challenge for the *802.11* standard is the range of the wireless signal. Typically, the wireless signal from a *802.11* device ranges upto 300 feet using a clear line-of-sight. The signal suffers greatly from physical obstacles and distance between devices. In order to provide a large-scale wireless network, an Internet service provider would need to set up a lot of access points so that users can stay connected to the network as they move from one location to another. To alleviate this problem, a new standard called WiMAX [70, 74] has been developed. WiMAX stands for Worldwide interoperability for Microwave Access and is a wireless metropolitan-area network technology that provides interoperable broadband wireless access to users. The service area of WiMAX extends upto 50 km and allows users to get network connectivity without the need for direct line-of-sight with the access point. Also, the throughput provided by WiMAX is as high as 75 Mbps.

Due to the computation and communication limitation of the nodes in a wireless network, there have been considerable developments in the field of energy-efficient MAC protocols. These protocols aim to save energy by periodically setting the interface to sleep, thus conserving the energy of the node and hence increasing the lifetime of the network [73]. Some

MAC protocols adaptively adjust the RTS-CTS handshake to mitigate the effect of hidden and exposed terminals.

In this section, we detailed the advances in the Medium Access Control technology for infrastructureless networks. The MAC layer provides fair and efficient access to the shared wireless medium. Advances in MAC protocols include TDMA, CDMA, FDMA and CSMA/CA and energy aware MAC protocols such as SMAC [73]. This research does not take into consideration the MAC protocol used by the wireless devices. Depending on the requirements of the network, any of the discussed MAC protocols for *802.11* can be used in conjunction with PILOT. PILOT is primarily a forwarding protocol and relies on the underlying MAC protocol to mitigate message collision and exposed terminal problems, as well as regulate energy consumption due to forwarding or receiving messages.

2.2 DISTRIBUTED HASH TABLES (DHTS)

Every DHT supports one basic operation *lookup (key).* Given a key, this function returns the identity of the location of the key. Hence, routing messages efficiently between the node that issues the query *lookup (key)* and the actual node that holds the key assumes paramount importance. The scalability and performance of a DHT is thus directly related to the routing algorithm employed. Examples of P2P systems that use a DHT are: Chord [63], Grid Location Service (GLS [30]), Pastry [54], Tapestry [76], FreeNet [9], Geographic Hash Table (GHT [62]) and Content Addressable Networks (CAN [52]). We will discuss CAN, GHT and GLS since they are most similar to this research.

CAN (Content Addressable Network) [52] provides a distributed, Internet-scale hash table. The key space in CAN is divided into a d-dimensional virtual Cartesian coordinate space. Each node contains a part of the distributed hash table, termed as a *zone.* Each node also holds information about adjacent nodes in the virtual network. In case of a request for a particular key, this request is routed to the node whose zone contains that key. Given a (key,value) pair, CAN maps the key to a point P in the co-ordinate system using a uniform hash function. The corresponding (key,value) pair is stored at the node that owns the zone in which P is located. To retrieve a value corresponding to a key, a node can apply the same hash function to obtain a point P. The request is routed to the zone which contains P. CAN uses a simple greedy forwarding algorithm by routing the request to the peer that is closest to the destination co-ordinates. The performance of CAN is closely related to the efficiency of the underlying routing protocol for the DHT.

The Grid location service (GLS) [30] provides distributed location information service in mobile ad-hoc networks. GLS combined with geographic forwarding can be used to achieve routing in the network. In GLS, nodes in the network use *location servers* distributed through the network to maintain their current location information. These location servers are not specifically designated and each node in the network acts as a location on behalf of some other node in the network. A node X recruits nodes that are "closest" to its own ID (least ID greater than X) in the ID space to act as its location servers. A node N trying to locate node X uses the same protocol as above to send a request to one of X's location servers. Upon receipt of this request from N, the location server forwards this request to X. Using the information about the position of N from its request node X uses geographic forwarding to initiate interaction with N.

GHT (Geographic Hash Table) [62] is a distributed data centric storage (DCS) system for sensor networks. GHT works by hashing keys into geographical co-ordinates and stores the (key,value) pair at the sensor node geographically closest to the hash of its key. GHT supports two basic operations: *put(k,v)*, that stores the value v that is associated with key k; *get(k)*, that retrieves the value that is associated with key k. This allows GHT to provide a hash-table based DCS interface for sensor networks that uniformly distributes load throughout the network. To ensure the availability of data due to node failures, GHT locally replicates the stored data. Also, GHT has a built-in mechanism which ensures that the key-value pairs are stored at the appropriate locations even after topology changes. GHT uses the Greedy Perimeter Stateless Routing (GPSR) [6], a geographic routing algorithm for wireless networks as its underlying routing algorithm.

In this section, we detailed the research work in Distributed Hash Table (DHT)-based systems that are similar to this thesis. CAN and GLS divide the network into zones based on a logical topology. In CAN, the overall logical co-ordinate space is dynamically partitioned among all the peers in the network. Each peer is responsible for a particular zone in the network. In GLS, the network is divided into a hierarchy of grids of increasing sizes. Both CAN and GLS use greedy forwarding to route messages from the source to the destination. Nodes use a scheme similar to GPS [13] to determine their geographic position. The logical topology is de-coupled from the physical topology and this can lead to inefficient routes in the network since a node that is "closer" in the logical topology may actually be quite far in terms of the physical topology. Similar to CAN and GLS, SARA also divides the network into zones, but unlike these schemes, this division is based on the physical topology of the network. Also, unlike CAN, when a new node arrives in the network, a zone is not split into two and this overhead is avoided. Mobility is also incorporated into the service architecture by using the concepts of *most likely residence* and *location vector* of a mobile node. Furthermore, the proposed research seamlessly integrates a power-aware data forwarding protocol into the architecture. This protocol uses the knowledge about the location of the source and the direction of the destination to forward traffic in a location-directed manner to reduce flooding in the network. The Geographic Hash Table (GHT) scheme is a distributed data centric algorithm that hashes keys into geographical co-ordinates and stores the (key,value) pair at the node geographically closest to the hash of its key. Similar to GHT, our approach also hashes keys into geographical co-ordinates, but instead of using *one* node in the network to hold the object information, a group of nodes are selected from within an area to hold the required information. Also, this information is stored in a manner such that only a fraction of it is necessary to *re-construct* it. The information about the registered entities is still available even after some nodes become mobile and leave the area where the information is registered.

2.3 ROUTING

Due to the lack of an existing infrastructure, routing in an ad-hoc network is of a very high priority. In an ad-hoc network, nodes cannot rely on base stations or access points to relay information from one node to another; routing between two nodes (assuming that the nodes are not in the direct vicinity of each other) would need to use the intermediary nodes for forwarding information from the source to the destination and vice versa. There are several routing protocols that have been suggested by researchers and these can be broadly categorized into four categories: pro-active routing protocols, re-active routing protocols, hybrid routing protocols and location-based routing protocols.

2.3.1 Pro-Active Routing Protocols

Pro-active routing protocols establish and maintain routes periodically. Routes are usually available before they are needed and route maintenance is of the highest priority with such protocols. The overhead can be very high for maintenance, but it can be offset if we have a network where communication between nodes is quite high. An example of a pro-active ad-hoc routing protocol is Destination Sequenced Distance Vector Routing (DSDV) [47].

DSDV is a pro-active routing protocol designed specifically for infrastructureless networks. Every node has its own copy of the routing table that lists all the available destinations and the number of hops it takes to reach each destination. Routing table information is exchanged periodically when every node broadcasts/multicasts its routing table along with a initiator-tagged sequence number. Each routing table entry in DSDV is assigned a sequence number so that nodes can easily distinguish between an old route (an invalid route) and a new route. On receipt of the routing table, the receiver node will check its routing table and replace all those entries whose sequence number is older than the one just received. If the sequence numbers are the same, then those with the smallest hop count are preferred.

2.3.2 Re-Active Routing Protocols

Re-active routing protocols establish routes as and when needed by a node. Route maintenance is very minimal and a route is usually maintained only for the lifetime of the connection. Re-active protocols reduce the load on each node and hence tend to be power conserving. These protocols perform well when the nodes do not communicate often. Otherwise, they tend to introduce a high latency due to the formation of routes. Examples of re-active routing protocols are: Ad-hoc On Demand Distance Vector Routing (AODV) [46] and Dynamic Source Routing (DSR) [34].

AODV is a re-active routing protocol for infrastructureless networks that creates and maintains routes as and when needed by the source nodes. When a source node needs a route to a destination for which it does not already have a route, it broadcasts a route request (RREQ) message. This broadcast is received by the other nodes who update their routing tables by inserting reverse pointers (along the backward direction). The RREQ is re-broadcast by nodes that receive them. Each RREQ is forwarded only once. If an RREQ is received again, then it is discarded. When the destination node receives the RREQ, a Request Reply (RREP) is initiated by the destination node. The path followed by the RREP to the source node is back along the same path that was followed by the initial RREQ.

DSR is a re-active routing protocol for infrastructureless networks that was developed to have a very low overhead and yet react quickly to changes in the topology of the network so as to be able to ensure successful delivery of packets. When a source node needs a route to a destination node, it broadcasts a route request. If the destination node hears the broadcast directly, it replies back, else all the nodes that have heard the route request broadcast do a limited broadcast of the request. A limited broadcast is when a node does re-broadcast a request, but discards it if it has already been processed. Any node that rebroadcasts the request adds itself to the path that the request has traversed. The reply follows the path to the source as it knows the route traversed by the request.

2.3.3 Hybrid Routing Protocols

Hybrid routing protocols use a combination of pro-active and re-active routing protocols. The highly mobile elements of the network are grouped into "clusters" and a pro-active scheme is used to route within a cluster, while routing between clusters is taken care off by a re-active scheme. Examples of hybrid routing protocols are: Zone Routing Protocol (ZRP) [25] and the (α, t)-Cluster [41, 1].

ZRP is a hybrid routing protocol that divides the network into overlapping zones [25]. Routing between and within zones are handled differently. Intra-Zone protocol (IARP) takes care of routing with a zone, while Inter-Zone protocol (IERP) deals with routing from a source node to a destination node not located in the same zone. Intra-Zone protocol can be any routing protocol that is pro-active. In this case, all nodes in a zone know the zone topology. Different zones can use different intra-zone routing protocols. In the Inter-Zone protocol a route to the destination node is found by first sending a Route-Request (RREQ) message to all the border nodes in the zone. This continues until the destination is found.

The (α, t)-Cluster routing protocol is a hybrid routing protocol that utilizes adaptive clustering to organize nodes into clusters in which the probability of path failure due to node movement can be bounded over time [41]. This metric allows for the dynamic balancing of the trade-offs according to temporal and spatial dynamics of the network. Intra cluster routing is done on a pro-active basis using a table driven pro-active routing algorithm. The (α, t)-Cluster framework is flexible and independent of the specific intra-cluster routing algorithm and hence, any pro-active routing algorithm designed for ad-hoc networks can be used for routing within a cluster. Inter cluster routing strategy tries to take advantage of the cluster topology and the intra-cluster routing tables. The Inter Cluster Routing Protocol (ICRP) is a fully reactive cluster based routing protocol that discovers and maintains routes on an on-demand basis. In ICRP, the central coordinator node for each cluster cooperates to control the route query process to avoid flooding the network.

2.3.4 Location-Based Routing protocols

Location-based routing protocols rely on the position of a node in space rather than on the topology of the network. These protocols take advantage of the fact that nodes in the network know their location (using a service similar to GPS [13]) and use this to optimize the protocol by sending the routing information in the direction of the destination rather than broadcasting it.

Location Aided Routing (LAR) [35] is a location-based routing protocol uses the location information of the nodes to improve the performance of routing protocols for infrastructure-less networks. LAR uses the concept of a *request zone*, an area within which the routing request is forwarded towards the destination. Nodes outside of this area drop the request. This helps in reducing the number of routing messages that are sent in the network. Two algorithms are provided to calculate the request zone. In the first algorithm, the source S calculates the request zone to be the smallest rectangle that contains S and the expected region of the destination D. This information is embedded in the route request and only nodes in the request zone forward this request. In the second scheme, S embeds its distance from D along with the position of D. An intermediate node forwards the route request only if it is "closer" to D as compared to S.

The Distance Routing Effect Algorithm for Mobility (DREAM) [56] is a location-based routing protocol in which each node in the network maintains a routing table that contains the location information about every other node in the network. A source S wishing to send a message to a destination D uses D's location information to send the message in the direction of D. This location information stored at nodes is updated as nodes move. The rate of updation is a function of the node's *mobility rate* (the speed of the node). The rate of location updates for a node increases as the speed of the node increases. Since the location updates are pro-active, this can lead to a large number of location update messages for a network consisting of highly mobile nodes.

Global Perimeter Stateless Routing (GPSR) [6] is a new kind of location-based routing algorithm that uses only the "local" neighboring information in its decision to forward the message to the next hop. The next hop node is chosen from a set of neighboring nodes by using a greedy forwarding scheme in a manner such that this node is the closest to the destination. This mechanism that the message is always forwarded closer to the destination. In the event when greedy forwarding fails (for example, it reaches a local maximum when there exists no neighboring node whose distance to the destination is closer), GPSR switches to recovery mode. In the recovery mode, the message is forwarded along the faces of the planar sub-graph, that are successively closer to the destination. This continues on until the message reaches a node closer to the destination than the source node, wherein greedy forwarding resumes.

In this section, we have described the research work related to routing protocols for infrastructureless networks with respect to pro-active, re-active, hybrid and location-based routing protocols. Pro-active routing protocols such as, DSDV [47] establish and maintain routes periodically. Routes are usually available before they are needed and route maintenance is of the highest priority. The overhead of these protocols though, can be very high since periodic updates require excessive bandwidth and processing, and frequently use scarce resources to maintain routes that may seldom be used. Re-active routing protocols such as, AODV [46] and DSR [34] establish routes as and when needed by a node. Route maintenance is very minimal and a route is usually maintained only for the lifetime of the connection. These protocols perform well when the nodes do not communicate often, otherwise they tend to introduce a high latency due to the formation of routes. Hybrid routing protocols such as, ZRP [25], (α, t)-Cluster [41, 1] use a combination of pro-active and re-active routing protocols. They abstract the highly mobile elements of the network into "clusters", and use a pro-active scheme to route within a cluster, while routing between clusters is taken care off by a re-active scheme. The disadvantage of all the above schemes is that none of them use the location information about the nodes in the network. To take advantage of the location information available, we have designed PILOT to be a location-driven routing protocol. Location-based routing protocols such as, LAR, DREAM and GPSR rely on the position of a node in space rather than on the topology of the network. These protocols use this location information to optimize the protocol by sending the routing information in the direction of the destination rather than broadcasting it. Due to the pro-active flooding of updates to a node's location information, both LAR and DREAM do not scale. PILOT differs from DREAM and LAR by not flooding the network with location updates; rather, messages are forwarded by intermediary nodes on a piece-meal basis (where the position of the destination is re-calculated and hence the direction of forwarding is changed to suit the direction of the destination). This leads to PILOT being more scalable when compared to DREAM and LAR. GPSR relies on the information about the neighboring node to be able to select the next hop to forward the message to. This information needs to be kept up-to-date for the correct performance of GPSR. PILOT, which is primarily a forwarding protocol forwards the message towards the destination and any node within this range (based on its probability of forwarding) picks up the message to forward this information. Also, unlike GPSR, which can hit a local maximum, PILOT has a built-in mechanism by which the probability of forwarding for intermediate nodes increases over time, thus ensuring that one of the nodes does indeed forward the message.

2.4 SERVICE DISCOVERY IN UBIQUITOUS ENVIRONMENTS

Service discovery provides an interface by which clients and servers discover each other and also the services that are available in the network. Service discovery for ubiquitous environments is still a very new area of research. There have been some protocols for service location and discovery that have been developed for LANs, namely: Service Location Protocol (*SLP*) [14] and Simple Service Discovery Protocol (*SSDP*) [75]. *SLP* relies on agents to search for and locate services in the network. The SLP framework consists of three agents, namely: : a *user agent*, a *service agent* and a *directory agent*. The user agent is used on behalf of a user to search for the required services. The service agent advertises the available services on behalf of a server. The directory agent collects the advertisements that are sent out by the service agent. *SSDP* uses HTTP UDP on the reserved local multicast address *239.255.255.250* and the *SSDP port* while searching for services. The ideas of SLP and SSDP cannot be directly applied to service discovery in ubiquitous environments due to their reliance on an existing network structure.

The Bluetooth Service Discovery Protocol (SDP) [24] is a mechanism by which devices enabled by Bluetooth discover the services provided by other Bluetooth devices. Each Bluetooth device maintains service records, each of which describes an available service. The service records contain information about the type of service, the necessary protocols for communicating with the service and location of appropriate documentation. To find services of interest, Bluetooth devices search the available space of services to find the service matching the required attributes. The Bluetooth Service Discovery Protocol is designed for small, personal-area networks with a small number of nodes.

Wibree [68] is a new interoperable low power radio technology for small devices such as, wrist watches, gaming and sports sensors that was developed by Nokia. Wibree allows these small battery operated devices to connect to host devices such as mobile phones, PDAs and personal computers. It is the first open technology offering such connectivity. In a press release dated June 12, 2007, it was announced that the Wibree forum was merging with the Bluetooth Special Interest Group (SIG) [69]. The fallout from this announcement meant that the Wibree specification will now become a part of the Bluetooth specification as an ultra low power Bluetooth technology. Wibree thus provides a natural low cost, low power extension to Bluetooth's Personal Area Network (PAN) by allowing for connectivity between small battery powered devices and Bluetooth enabled devices such as phones and PDAs.

Jini [65] is a service discovery protocol developed by Sun Microsystems is an agent-based service discovery protocol that is based on Java. A lookup service that is part of the service discovery framework is responsible for maintaining information about the services in the network. Services are registered with the lookup service by the service provider as and when they become available in the network. Clients wishing to avail of services in the network query the location service, which provides them with information about the service and also the means of contacting the server providing the required service.

Universal Plug and Play is a set of network protocols for service discovery developed by the Universal Plug and Play Forum [67]. The goal of UPnP is to allow for seamless integration of devices for the purpose of implementing small personal area networks (in home or office). UPnP device and service definitions and definitions are defined in XML and are built on open, Internet-based communication protocols such as: HTTP and SOAP. Service discovery requests are made on demand and services are advertised as and when necessary.

Konark [60] is a service discovery and delivery protocol for ad-hoc networks that is geared towards device independent services. The Konark architecture is operating system and programming language independent since it is built above the network layer and uses XML. Each device in Konark is treated as a client as well as a server and include a Konark application that facilitates service advertising and discovery along with an SDP manager and registry that maintain service information about services offered by peers in the network. In addition to this, each Konark device runs a micro-HTTP server that acts as a handler for service delivery requests. Service information in the registry is stored in the form of a service tree with the leaf nodes representing service names while the root and internal nodes represent service types. Services are advertised periodically to the entire network. To discover a service, a client sends a discovery request to its local multicast group.

Ekta [23] integrates DHTs (akin to peer-to-peer systems such as [63, 76, 53, 9, 42, 55]) into MANETs. This provides an efficient architecture for constructing distributed applications and services in MANETs. Two different approaches are studied: a layered approach that directly overlays a DHT on top of the existing multi-hop routing protocol used for the infrastructureless network and an approach that directly integrates the DHT with the multi-hop routing at the network layer, thus maximizing the advantages created by their interaction. The layered approach is implemented by layering Pastry [54] on top of DSR [34], while the integrated approach integrates Pastry and DSR at the network layer.

Twins [72] provides an architecture for addressing and locating nodes in large networks. Twins uses a DHT-based architecture for location management in self-organizing networks. Twins employs a dual addressing space (logical and geographical) to achieve this purpose. This multidimensional space is a strict mathematical representation of the network geographic space. The logical space is partitioned among the network nodes and is used for locating nodes in the network based on the partitions assigned to the nodes. Geographic forwarding is achieved by using a simple forwarding mechanism that assumes that all the nodes know their location information using a mechanism such as GPS [13]. The logical space is mapped onto the network geographic space by using a multi-to-one dimensional mapping that is based on the concept of Hilbert space-filling curves.

A cross layer approach to service discovery in MANETs is provided in [4]. This provides a routing protocol independent library, called the Service Discovery Library (SDL), that works at the user level and is responsible for service registration and discover and a Routing Layer Driver (RLD) that closely interacts with the routing protocol to keep track of topology changes and also to propagate service discovery requests. Service registration and discovery matching are based on simple string based comparison. Two prototypes are presented, namely: CL_{dsr}, which uses DSR [34] as its underlying routing protocol, and CL_{dsdv}, that uses DSDV [47] as its underlying routing protocol.

GSD (Group Service Discovery) [11] uses the Web Ontology Language (OWL) [43] to describe service and resources that are available with the network nodes in an ad-hoc network environment. Based on the service description, the available services are organized into a hierarchy of “groups”. Services are advertised periodically to other nodes in the network. Apart from advertising its own services, a node also advertises the service groups that it has seen around its neighborhood. When a node requires a service, it first initiates a service request locally by querying its 1-hop neighbors. In the event when a service request is not resolved by a node's neighbors, this request is selectively forwarded to other nodes who belong to the same service group in order to increase the probability of finding information about the requested service.

A scalable service discovery protocol for MANETs is provided in [18]. Service registration and discovery services are provided by a set of co-operating directories that form a virtual network. These directories are formed by network nodes that maintain a cache of the available web services within their vicinity. To allow for global discovery of services, each directory summarizes its content and periodically broadcasts this information to the other directories in the network. 2-hops bordercasting is used for the purpose of reducing the network traffic. To reduce the size of the directory summaries, Bloom filters [7] are used. In order to perform a service discovery, a client node queries its local directory. In the event that the information is not available in the local directory, the query is forwarded to the other appropriate directories in the network.

In this section we have described in detail the research advances related to service discovery in infrastructureless networks. SSDP [75] and SLP [14] provide service location and discovery in LANs and they cannot be directly applied to service discovery in ubiquitous environments due to their reliance on an existing network structure. The Bluetooth Service Discovery Protocol [24] along with Wibree [68] is designed for small, personal-area networks consisting of a small number of battery powered devices as well as Bluetooth enabled devices such as phones and PDAs. Bluetooth technology cannot be applied to large-scale ubiquitous environments. Jini [65] is an agent-based service discovery protocol responsible for maintaining information about the services in the network. The ideas of Jini cannot be directly applied to service discovery in ubiquitous environments due to its reliance on a centralized location server. Another deterrent to using Jini is the requirement that devices in the network be Java enabled. Furthermore, Jini uses a RPC mechanism (the Java Remove Method Invocation (RMI)) that consumes a lot of network resources. UPnP is a service discovery protocol developed by the Universal Plug and Play Forum [67]. UPnP uses a lot of bandwidth due to duplicate message and hence is not a suitable solution for resource constrained ubiquitous environments. Konark [60] is a service discovery and delivery protocol for ad-hoc networks that is geared towards devices independent services. Though Konark provides an architecture for service discovery in infrastructureless networks, it is expensive in terms of communication costs due to the number of messages sent and hence does not consider energy constraints or delay. Our service architecture, SARA contains an integrated power-aware message forwarding protocol that takes into consideration the residual energy level at an intermediary node while forwarding messages towards the destination, thus maximizing network lifetime. Ekta [23] provides an efficient architecture for constructing distributed applications and services in MANETs. Our service architecture, SARA differs from Ekta, since it does not use Pastry for the purpose of its DHT. The Virtual-DHT is constructed in a manner such that it is able to take advantage of the location information provided in the network. Our DHT scheme is more lightweight when compared to Pastry and also takes node mobility into consideration. SARA also contains an integrated power-aware message forwarding algorithm that is more efficient than DSR since it uses location information provided in the network. Twins [72] provides an architecture for addressing and locating nodes in large infrastructureless networks. Our approach differs from TWINS, in that we do not use multiple address spaces. This overhead of mapping one space onto another is avoided by directly mapping the DHT onto the physical structure of the network. Node mobility is also incorporated into the framework by using the *location vector* of the mobile node. [11] uses a cross layer approach to service discovery in MANETs. Though the cross layer approach is similar to our research, the authors do not consider node mobility and rely on the underlying routing protocol to route traffic. We incorporate node mobility into SARA by using the *location vector* of the mobile node. Both GSD [11] and [18] provide service discovery in MANETs by using local broadcasts for service advertisements. In the event that a service is not available locally or with the local directory as in the case of [18], the request is forwarded through the network or to the next directory. These schemes do not take node mobility into consideration. The mobility of nodes causes frequent service advertisements which results in an implosion of messages in the network that uses valuable network resources. The schemes described in [33, 71, 48] focus on resource discovery in MANETs. Similar to these research works, our scheme also provides support for resource discovery in ubiquitous environments, but uses the concept of virtual residences to bootstrap the resource discovery process. Our architecture also contains a power-aware data forwarding scheme.

2.4.1 Home Region/Agent based schemes

Mobile IP [17] was developed to facilitate mobile computing. The main idea was to allow users the ability to take their computing environment along with them and continue their work without having to change their configurations. The only requirement was that the user have a connection to the Internet in the new location. The mobility of the users is handled by allowing the mobile node to have two IP addresses: a *home-address* and a *care-of address*. The home-address of the node is the fixed IP address of the node, while the care-of address is the address that the node acquires at its new location. Traffic intended for the mobile node (that was held up) is forwarded by its *home agent*, upon receipt of the node's care-of address. While mobile IP solves the problem arising due to the mobility of the nodes in the network; services that are provided are not continuous, owing to the fact that the node has to register its care-of address with its home-agent.

[27, 59, 26, 57] use the concept of home agents (or) home regions. Each node in the network is mapped to an area (using a hash function) in the network that is designated as its home region. The home region holds the location information about the mobile nodes which map to this location. A mobile node updates its location information by sending updates to its home region. We differ from the above schemes by not updating a node's home region. Instead, we used the *location vector* of a node to leave behind it's mobility information using which other nodes can locate it.

2.5 SUMMARY

This chapter presented an overview of the research related to this thesis in the areas of: Medium Access Control (MAC), Distributed Hash Tables (DHT)s, routing and service discovery in ubiquitous environments. Section 2.1 provided a background and listed the advances in Medium Access Control technology. The available body of literature ranges from the traditional CSMA/CA MAC protocols to energy efficient MAC protocols such as, SMAC [73]. Section 2.2 detailed the research work related to Distributed Hash Table (DHT)-based systems. In this section, a brief overview of DHTs was provided along with examples DHTs such as, CAN, GLS and GHT. Though our work is similar to GLS, CAN and GHT in terms of mapping services to zones, in our approach the zones are divided based on the physical topology. Also, our work seamlessly integrates a power-aware message forwarding protocol into the framework. Section 2.3 described in detail the research work related to routing protocols for infrastructureless networks. In this section, the body of related work was broken up into into different categories, namely: pro-active, re-active, hybrid and location-based routing protocols. Examples were provided for each of these different kinds of routing protocols. Section 2.4 concludes this chapter by providing in detail the research advances related to service discovery in infrastructureless networks. In this section, a brief overview of infrastructured service discovery protocols, such as SLP and SSDP was provided. The rest of the section was dedicated to the discussion of existing service discovery protocols for infrastructureless networks.

3.0 SARA: A SERVICE ARCHITECTURE FOR RESOURCE AWARE UBIQUITOUS ENVIRONMENTS

In this chapter, we present SARA, a novel resource and location aware framework to support large-scale deployment of service and applications in resource aware ubiquitous environments. We begin by examining the characteristics of the service architecture and then present the components of SARA. Section 3.2 details the building blocks used in SARA while Section 3.3 details the services of SARA and provides the algorithms used in SARA for virtual registry creation and management, location registration, object registration and discovery, mobility management and node interaction.

3.1 OVERVIEW

The goal of SARA is to address the fundamental design issues of a service infrastructure for ubiquitous environments and provide a comprehensive solution which takes into consideration node mobility and resource constraints. To this end, there are several challenges that must be addressed in order to develop such an architecture. These challenges are related to the development of several capabilities necessary to support node interaction in SARA. These capabilities include: object registration, object discovery, mobility management and data dissemination. SARA is divided into four components (as shown in Figure 3.1) with each component responsible for one of the above capabilities. Each of the components export certain pre-specified interfaces which the other components use to interact with it. This allows for easy and flexible access within and between components.

SARA is visualized as an underlying system that exports certain well-defined primitives

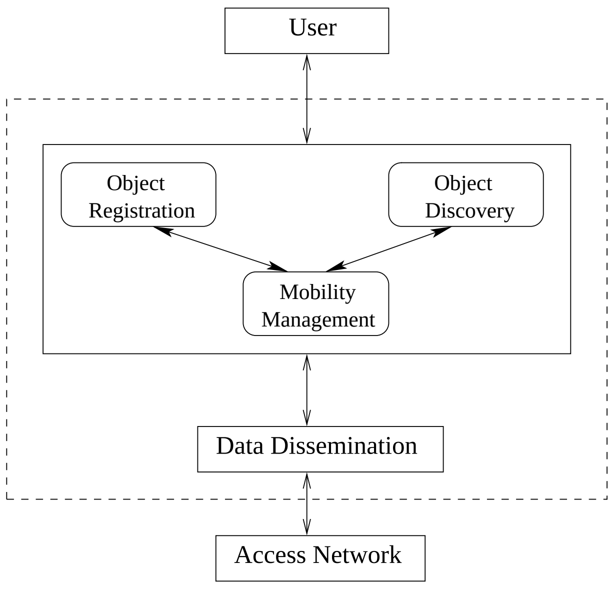


Figure 3.1: SARA Architectural Components

that allow users to register or discover objects in the network. A user wishing to utilize the system can choose to perform either an object registration or an object discovery by using the exported interface. Unlike traditional wireline networks, node mobility is a factor in wireless networks and has to be accounted for. In a network like this, not only do we need to account for node mobility, we also have to take into account the issues with respect to change in network density and network eccentricity. As in real life, networks are rarely uniformly distributed. An efficient way to build and sustain the network is important, for most nodes in the network have limited energy resources. To account for node mobility, the registration and discovery components use the mobility management component to be able to locate a mobile node. The underlying data dissemination protocol is then used to route application or control traffic to the intended destination.

3.1.1 Object Registration

Object registration is the process by which a node, containing a set of objects that are of interest to other network nodes, registers its objects with the network. It is necessary for the node to locate the node or nodes in the network with which it must register this information. This information typically contains the object id, its name and its relevant attributes. A user performing an object registration uses the exported interface to send and exchange information with this component.

The lack of a fixed infrastructure and the mobility of nodes in the network makes object registration challenging. Due to the lack of a fixed infrastructure, there is no central directory available to register services. To facilitate registration services in such a network, the schemes developed must be distributed and be immune to the mobility of nodes in the network. Also, node mobility makes it difficult to locate the node that owns the required object. To allow for locating mobile nodes in the network, a node must also register its location information while registering its objects, to facilitate node interaction. The schemes developed must also take into consideration the resource constrained environment.

3.1.2 Object Discovery

Object discovery is the mechanism by which nodes in the network discover the objects of interest that are available in the network. Typically, nodes try and locate the node or nodes with whom the object is registered (registry nodes) and queries for the object of interest. If a match occurs, the registry nodes reply to the query with a list of nodes in the network that *own* that object. Using this information, the node can choose to establish a connection with any one of the nodes on the list to request the object. A user performing an object discovery uses the pre-defined interface exported by this component to interact with it.

The challenges faced for object discovery in infrastructureless networks are very similar to those faced by object registration. To locate a mobile node (after a successful query for an object), the discovery component makes use of the mobility management component to locate the required mobile node.

3.1.3 Mobility Management

Mobility management is an important aspect of this framework, since nodes in an infrastructureless network can be mobile. An efficient mechanism must be in place whereby nodes in the network can locate, with high probability, the mobile node. The scheme developed to locate a mobile node must be resource aware. Due to the mobility of nodes in the network, both the object registration and discovery components need to interact with this component. This interaction is achieved through the use of the exported primitives of each component.

To track mobile nodes, the network requires information to be stored that clearly goes beyond the typical information stored for a service architecture. In addition to the object information, the mobility information of a node must also be stored in order to facilitate interaction between nodes. This node information, however, changes dynamically, as the node moves from one location to another. Efficient mechanisms must, therefore be in place to update this information as nodes move.

3.1.4 Data Dissemination

Data dissemination is the mechanism by which application and control traffic reaches its intended destination. The strategy used to forward traffic efficiently must take into consideration the time-varying dynamics of the network, node mobility and power-consumption. The tradeoffs between these important design factors and network characteristics must be recognized and alternatives carefully evaluated. This component is used by the registration, discovery and mobility management components to send and retrieve traffic from the network. This component uses the primitives provided by the underlying network layer to send and receive packets from the network and exports its own interfaces to the layers above it.

To limit flooding in the network, the data dissemination protocol uses the knowledge about the location of the source and destination to forward traffic in a directional manner towards the destination. The intermediary nodes that forward traffic are chosen based on a priority-based scheme that imposes a priority on the neighboring nodes in a way, such that nodes which are more in line with the direction of the destination have higher priority to forward the message. This reduces the delay that traffic suffers on its way to the destination. Chapter 4 talks in detail about PILOT, the data dissemination protocol used in this architecture.

3.2 SARA BUILDING BLOCKS

This section details the building blocks of SARA. The main building blocks of SARA are *virtual registries* and the *virtual anchor forwarding path*. A virtual registry in SARA represents a physical area to which objects are mapped and it acts as a “reference point” between the nodes that provide objects and the nodes that request objects. The virtual anchor forwarding path is a global structure that spans the network service area, and consists of a set of landmark points where each landmark represents a physical location in the network. The virtual anchor forwarding path determines the path to follow to locate the next available virtual registry.

3.2.1 Virtual Registries (*VR*s)

Consider a ubiquitous networking environment covering a specific geographical area, denoted by Λ. The network service area is sub-divided into regions according to the coverage of network nodes by using a Voronoi Diagram. The network consists of a collection of entities ($\mathcal{E}$), $\mathcal{E} = \{E_i^t, i = 1, ..., M; t = 1, ..., K\}$, where t: entity type and i: number of entities of each type. Types of entities in the network include nodes (users) and objects. Each entity is associated with a *unique id* and *metadata*. Entity metadata consists of the type of entity, its attributes and information about the node that owns the entity. Entities are mapped into the network by using a Distributed Hash Table (DHT) that uses geographic mapping for its DHT as opposed to node mapping.

A node (user) in the network is characterized by its *unique id*, its *most likely residence (MLR)*, its *location vector (LV)* and the collections of objects owned by it. The *MLR* of a node is the physical location where the node is expected to spend most of its time and is used as a *congregation* point by nodes to contact other nodes. The location vector (LV) of a mobile node is a dynamic time-dependent vector that represents the most likely physical location of the node at a given time, thus reflecting user activity. The location vector of a node, N, is represented as: $LV_N = \{L_N(t_1), L_N(t_2), ..., L_N(t_n)\}$, where $L_N(t_1)$ is the most likely physical location of node N during time period t_1. The location vector is generated by using a user-defined schedule that comprises of dates, times, and events. The location vector typically depicts the everyday behavior of the user. For example, users can specify that during a specific time of day they are available at home, at work or out shopping. However, incidental events may occur that impact this location vector. These changes are conveyed to the user schedule and the location vector is appropriately updated to capture the specific events that occur that day and reflect the new user activity. The user-defined schedule can easily be composed and modified by using a calendar based system.

The collection of objects owned by node N, is represented as a list: $O^N = \{O_i^N, i = 1...\textit{number of objects}\}$. Objects in the network are of two types: pre-existent and on-demand. Pre-existent objects are those that are carried by mobile devices belonging to users. Types of such objects include: documents, music, video, medical information, maps. On-demand objects are objects that are created as and when necessary. Types of such objects include data gathered as a result of a query. Each object is mapped into geographical coordinates in the network, Λ by using a Distributed Hash Table. This mapping is achieved using a dual-valued hash function $H()$, such as $H(O_i) = [x, y]$. The mobile nodes in the virtual registry containing $[x, y]$ are responsible for maintaining and managing information related to O_i. This responsibility includes resolving requests regarding the current location of the node which owns the object along with any other relevant attributes.

In order to balance the distribution of service information among the network zones, the hash function, $H()$, must be uniform [12]. Furthermore, it is desirable that the computational cost and collision of $H()$ be minimal. To meet these requirements, the function $H()$ is defined as follows:

$$H(O_i) = \begin{cases} H^x(O_i) = \lfloor L * (O_i * A - \lfloor O_i * A \rfloor) \rfloor \\ H^y(O_i) = \lfloor M * (O_i * A - \lfloor O_i * A \rfloor) \rfloor, \end{cases} \tag{3.1}$$

where: O_i is a *m-bit* object id, $0 < A < 1$ is a constant, Λ is a network area of size MxN and $H^x()$ and $H^y()$ are uniform hash functions. It is shown in [12] that a good choice for A is: $(\sqrt{5} - 1)/(2)$.

The network service area, Λ consists of several virtual registries. A virtual registry (VR) is a dynamically created administrative domain that enables entity registration and discovery. Information is managed in a virtual registry by a collection of mobile nodes within that area. A virtual registry is characterized by its id and its member nodes. The extent of a virtual registry is such that it encompasses at least $\mathcal{K}$ threshold neighboring nodes. This is to ensure that there exists enough number of nodes in the virtual registry to register and discover entity information. The process of creating a virtual registry and managing it is detailed in Section 3.3.1.

3.2.2 Virtual Anchor Forwarding Path

The Virtual Anchor Forwarding Path (VAFP) provides a path that spans the network service area, Λ. This path is created off of a set of landmark points, where each landmark point represents a physical location in the network. These landmark points determine the path to follow to locate the next virtual registry. To locate the next available virtual registry, the nearest landmark is located and the *VAFP* is traversed in the direction indicated. Such a structure ensures that there is no data loss in the event of a virtual registry disbanding or when there are no virtual registries at the registering point. An example of such a structure is shown in Fig. 3.2 and the algorithm to traverse the *VAFP* is provided in Algorithm 1.

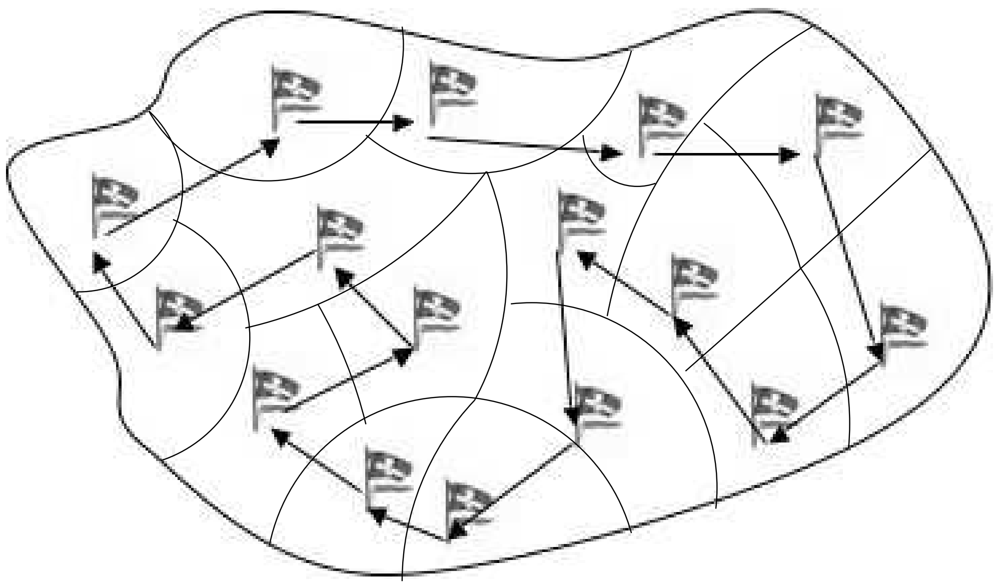


Figure 3.2: Virtual Anchor Forwarding Path

Algorithm 1 Forwarding along the *VAFP*

```
VAFP-FWD($x$, $y$, $msg$)
Input: $x$, $y$, $msg$
Output: Result
1: Locate closest landmark ($B_i$) to [$x$, $y$] along the $VAFP$
2: Forward $msg$ towards $B_i$
3: Traverse the $VAFP$
4: Register (or request information) with (from) the first available registry
5: if $VAFP$ traversed completely then
6:     Drop $msg$
7: end if
8: return
```

3.3 SARA SERVICES

This section details the services provided in SARA, with respect to virtual registry creation and management, location vector registration, object registration and discovery and mobile node location. Registration and discovery are achieved by hashing the *entity id* to obtain the physical co-ordinates of a *point* (P) with the network service area. The set of mobile nodes in the virtual registry containing P assume the responsibility of maintaining information about the object. The basic *design principle* for our scheme is to use *geographical* mapping for the hashing as opposed to *node* mapping since nodes are mobile.

3.3.1 Virtual Registry Creation and Management

Virtual registries are dynamically created in a region when there are at least $\mathcal{K}$ threshold nodes available. Consider the scenario when a node starts up in its most likely residence. Initially, it tries to register itself to the network by registering its location vector and its objects. While registering, a node also indicates if a virtual registry exists in the region where its *MLR* is located or not. Information about the regions where a registry does not exist is exchanged asynchronously amongst the available registries in the network through the use of the *VAFP*. The information exchanged contains the location of the region and the number of nodes in the region that declare this region to be their *MLR*. Once the number of such nodes crosses the threshold value, $\mathcal{K}$, a message indicating to the nodes to form a virtual registry is sent to the corresponding region. During network startup, the first node to register itself with the network will not be able to locate any registry in the network and hence will form a registry of its own that acts as a "temporary" registry that bootstraps the registry creation process. The process of creating a new virtual registry is shown in Algorithm 2.

Algorithm 2 Virtual Registry creation

```
VR-CREATE($\mathcal{K}$)
Input: $\mathcal{K}$
Output: Result
1: LV-REGISTER($N_{id}$, $H$, $LV_N$)
2: Register information about registry status
3: Exchange information about regions that do not have a registry using the VAFP
4: Update the number of nodes in a region based on above information
5: if Number of nodes in a region $\geq$ $\mathcal{K}$ then
6:     Send message to region to form virtual registry
7: end if
8: return
```

The *id* of the newly created virtual registry is the $[x, y]$ physical location of the landmark that exists in this region. Area of VR coverage is the area of the region in which this registry exists. The virtual registry registers information about all entities whose hash value falls within its coverage area. After a new VR is created, the *VAFP* is traversed completely to find the subsequent VRs along the *VAFP* to collect any information pertaining to it and as well as its preceding areas.

The information in a virtual registry is managed in a manner such that k out of N

(number of nodes) nodes are enough to re-construct the original entity information. This is to ensure that the information about the entities registered at this registry is still available even when a few nodes decide to leave the registry ([39] Defines a key encoding scheme that can be used for this purpose). In a virtual registry, every node is aware of every other node and nodes update each other through the use of *HELLO* messages. If the number of nodes belonging to a virtual registry is close to falling below the threshold, $\mathcal{K}$, the registry attempts to recruit new nodes to maintain the threshold. If the number of available nodes in the registry does go below $\mathcal{K}$, the virtual registry is dissolved and the existing information about the entities is forwarded along the *VAFP* to the next available registry.

3.3.2 Location Vector Registration

Before registering its objects, a node registers its location vector with the network. The advantage with this is that objects need not be re-registered even in the event of a change in the node's location vector. It only needs to "correct" its location vector at the registry where it registered. To register its information, a node, N performs a hash on its id (N_{id}) to get a hash value. This hash value maps to the physical co-ordinates of a point (P) within the network service area. Using directional routing, N sends information in the direction of P to register its *LV*. The set of nodes in virtual registry containing P register the location vector of N. If there exists no registry containing P, the information is forwarded to the next registry along the *VAFP*. Algorithm 3 details the process by which a mobile node N registers its location vector with the network (H is the uniform hash function).

The duration for which a location vector is registered depends on the mobility of the node. An almost static node will be able to provide the location vector for a longer duration as compared to a highly mobile node.

3.3.3 Object Registration

A node N, that owns a collection of objects must register its objects with the network in order for other nodes in the network to locate its objects. A node registers its objects one at a time by first hashing the *object id* to obtain a *hash value*. This hash value maps to

Algorithm 3 Location Vector Registration

```
LV-REGISTER($N_{id}$, $H$, $LV_N$)
Input: $N_{id}$, $H$, $LV_N$
Output: Result
 1: Calculate $H(N_{id}) = [x_i, y_i]$
 2: Let $[x_i, y_i]$ be the physical co-ordinates of a point $P$ within $\Lambda$
 3: Compose message ($msg$) consisting of $N_{id}$ and its location vector, $LV_N$
 4: Use directional routing to send $msg$ towards $P$
 5: if $\exists$ VR containing $P$ then
 6:     Register $LV_N$ and acknowledge
 7: else
 8:     VAFP-FWD($x_i$, $y_i$, $msg$)
 9: end if
10: if $LV_N$ not Registered then
11:     Try again after a timeout
12: end if
13: return
```

the physical co-ordinate of a point (P) within the network service area. Using directional routing, N sends information in the direction of P to register the object. The information consists of the object id and its metadata. The set of nodes in virtual registry containing P register the object information. If there exists no registry containing P, the information is forwarded to the next registry along the *VAFP*. Algorithm 4 details the process by which a mobile node N registers information relevant to object O_i^N.

Algorithm 4 Object Registration

```
OBJ-REGISTER($O_i^N$, $H$)
Input: $O_i^N$, $H$
Output: Result
 1: Calculate $H(O_i^N) = [x_i, y_i]$
 2: Let $[x_i, y_i]$ be the physical co-ordinates of a point $P$ within $\Lambda$
 3: Compose message ($msg$) consisting of $O_i^N$ and its metadata
 4: Use directional routing to send $msg$ towards $P$
 5: if $\exists$ VR containing $P$ then
 6:     Register $O_i^N$ and acknowledge
 7: else
 8:     VAFP-FWD($x_i$, $y_i$, $msg$)
 9: end if
10: if $O_i^N$ not Registered then
11:     Try again after a timeout
12: end if
13: return
```

As shown in Algorithm 5, node N repeatedly uses Algorithm 4 to register multiple objects.

Algorithm 5 Multiple Object Registration

```
MULT-OBJ-REGISTER($O^N$, $H$)
Input: $O^N$, $H$
Output: Result
1: i = 0
2: while i < number of objects do
3:     OBJ-REGISTER($O_i^N$, $H$)
4:     i++
5: end while
6: return
```

3.3.4 Object Discovery

Object discovery follows a procedure similar to object registration. A node D, that wishes to locate the object(s) of interest in the network first performs a hash on the the *object id* to obtain a *hash value*. This hash value maps to the physical co-ordinate of a point (P) within the network service area. Using directional routing, D sends a request in the direction of P for information about the object. The nodes in the registry containing P reply with the object information. If there exists no registry containing P, the request is forwarded to the next registry along the *VAFP*. The object information includes: object id and object metadata. Using the object metadata, node D obtains the virtual registry with which the node that owns the object (N) registered its location vector (LV_N). Let VR_N be the virtual registry with which node N registered its location vector. D sends a message to VR_N to procure LV_N. Using LV_N, node D can now initiate communication with node N. Algorithm 6 details the process by which a mobile node D discovers information relevant to object O_i in SARA.

It should be noted that in the above example, we assumed that the reply about O_i contained information about only one node that owns it. This reply typically consists of a list of nodes that own O_i. Node D can choose to establish connection with any one of the

Algorithm 6 Object Discovery

```
OBJ-DISCOVER($O_i$, $H$)
Input: $O_i$, $H$
Output: Result
 1: Calculate $H(O_i) = [x_i, y_i]$
 2: Let $[x_i, y_i]$ be the physical co-ordinates of a point $P$ within $\Lambda$
 3: Use directional routing to send request towards $P$
 4: if $\exists$ VR containing $P$ then
 5:     Reply with $O_i$ and its metadata
 6: else
 7:     VAFP-FWD($x_i$, $y_i$, request)
 8: end if
 9: if Response received before timeout then
10:     Send request for $LV_N$ towards $VR_N$
11:     if $VR_N$ exists then
12:         $VR_N$ replies with $LV_N$
13:         Establish connection with $N$
14:     else
15:         VAFP-FWD($x_i$, $y_i$, request for $LV_N$)
16:     end if
17: else
18:     Try again after timeout
19: end if
20: return
```

nodes on that list. This decision can be made based upon several factors: prior knowledge of node, distance to virtual registry and actual distance to the node.

3.3.5 Mobile Node Location

Mobile nodes in the network register their location vectors to facilitate node interaction. In the event that a node deviates from its location vector, it re-registers its location vector. Consider the scenario when a node, A becomes mobile and leaves its current location for a brief period. It is quite expensive to re-register the new mobility pattern every time a node moves. Unless the deviation from the current location is for a long duration or a repeated event, it is more efficient for a node to provide a temporary *mobility profile*. Node A has knowledge about its intended destination and hence its direction and speed of travel. Node A leaves behind this information in the form of a *mobility profile* with select *proxy* nodes that act as the *Mobility Profile Management Base (MPMB)*. Nodes that wish to contact A can predict the new location of A based on its *mobility profile* and the `elapsed` time since this information was provided [2]. Algorithm 7 details the steps using which a node C establishes contact with node A. The components of the *mobility profile* in Algorithm 7 are: t_0: starting time, $V(t_0)$: expected starting speed, $D(t_0)$: expected initial direction, $P_v(t)$: Predictor for speed after t time units since A's departure, $P_d(t)$: Predictor for direction after t time units since A's departure.

To allow for more flexible node mobility and accommodate random mobility, a node also sends back *corrections* with regards to its *mobility profile* to the *MPMB*. The *MPMB* updates the mobility profile of A to reflect this change. The mobile node A needs to find the set of *proxy* nodes that form the *MPMB* in its virtual registry to leave its *mobility profile* with. To recruit *proxy* nodes, node A sends out a broadcast message within its registry and waits for replies from the other nodes. The *mobility profile* is encoded in a manner such that k out of N (number of replies) fragments are enough to re-construct the original *profile*. This is to ensure that the mobility profile is still available even when a few proxy nodes decide to leave the registry ([39] Defines a key encoding scheme that can be used for this purpose). The steps involved in building the *MPMB* are described in Algorithm 8.

Algorithm 7 Handling node mobility

```
NODE-MOBILITY()
Input: Void
Output: Result
 1: C receives $LV_A$ from $VR_A$
 2: Using directional routing, $C$ sends message towards $A$'s current location
 3: if $A$ is in its correct location then
 4:    Connection established between $C$ and $A$
 5: else
 6:    The MPMB replies with the mobility profile of $A$, a metric:
       $[t_0, V(t_0), D(t_0), P_V(t), P_D(t)]$
 7: end if
 8: $C$ uses the mobility profile of $A$ to determine with high probability its current position
    and sends messages in this direction
 9: $A$ upon receiving the messages by $C$ acknowledges it and initiates a conversation with $C$
10: return
```

Algorithm 8 Building the MPMB

```
BUILD-MPMB($k$)
Input: k
Output: Result
 1: Broadcast request MPMB_FORM
 2: Let number of replies be $N$
 3: Encode mobility profile such that $k$ out of $N$ fragments are enough to re-construct the
    original profile
 4: Send encoded parts to the $N$ nodes
 5: return
```

3.4 SUMMARY

This chapter presented the complete specification of SARA, a novel resource and location aware framework to support large-scale deployment of service and applications in resource aware ubiquitous environments. First, the requirements of a service architecture were detailed in Section 3.1. Each of the components required for a service architecture were elaborated upon. Section 3.2 detailed the building blocks used in SARA while Section 3.3 detailed the services used in and provided the algorithms used in SARA for registry creation and management, object registration, discovery and mobile node location services.

The basic tenet of SARA revolves around the concepts of *virtual registries*, *most likely residence* and *location vector*. A virtual registry is a dynamically created administrative domain that enables object registration and discovery. The extent of a virtual registry is such that it encompasses at least $\mathcal{K}$ (threshold) nodes. The information in a registry is maintained by its member nodes. The *most likely residence* of a node is the physical area where the node is likely to be located most of the time and is used as a *congregation* point by nodes to contact other nodes. In the case of a mobile node, the node also registers its *location vector*. The location vector is a dynamic time-dependent vector that represents the physical location of the node at a given time, thus reflecting user activity. The *primary* advantage of this approach is that each node can choose to provide its *own* mobility prediction model, which it deems to be most appropriate to its current activity, rather than using a network-wide model which may not be applicable to specific itineraries and situations.

Object registration and discovery are achieved by hashing the *object id* to obtain the physical co-ordinates of a *point* (P) within the network service area. The set of mobile nodes in the virtual registry containing P assume the responsibility of maintaining information about the object. The basic *design principle* for our scheme is to use *geographical* mapping for the hashing as opposed to *node* mapping since nodes are mobile. While bootstrapping, a node only needs to know the hash function that is used to register and locate objects in the network. A sensitivity analysis of SARA through simulation is provided in chapter 7. SARA is also integrated with a power aware message forwarding protocol. This protocol, called PILOT is presented in chapter 4.

4.0 PILOT: A POWER-AWARE LOCATION DRIVEN TRAFFIC FORWARDING ALGORITHM

In this chapter, we present, PILOT, a data-forwarding algorithm for ubiquitous environments that is used by the service architecture, SARA. PILOT forwards traffic in a power-aware location-directed manner. We start by providing an overview of PILOT and detail the characteristics of PILOT in Section 4.1. Section 4.2 describes in detail the protocols used to make the traffic forwarding algorithm power-aware by tying the *probability of forwarding* to the *residual power* at an intermediary node and Section 4.3 concludes this chapter by providing a probabilistic analysis of PILOT.

4.1 OVERVIEW

In this section, we provide a brief overview of PILOT and its characteristics. To limit flooding in the network, PILOT uses the knowledge about the location of the source and the direction of the destination to forward traffic in a truncated cone-shaped manner towards the destination. The intermediary nodes that forward traffic are chosen by using a priority-based scheme that imposes a priority on the neighboring nodes in a way, such that nodes which are more in line with the direction of the destination have higher priority to forward the message. This reduces the delay that traffic suffers on its way towards the destination. As traffic progresses towards the destination, the highest priority node responsible for forwarding traffic calculates a new cone and re-iterates the process.

PILOT is also power-aware and it ties the *probability of forwarding* of an intermediate node to its residual energy level, to maximize network lifetime. To reduce computational costs while calculating the *probability of forwarding* with respect to the residual energy level, a power matrix that is calculated offline is used.

4.1.1 PILOT Characteristics

Let us consider a scenario when a node S tries to contact another node D. Using the knowledge about the position of D and its *expected* direction and speed, node S tries to route the traffic to node D. To reduce flooding in the network, the traffic is limited to a truncated, cone-shaped region whose central-line is directed towards the direction of D, as shown in Figure 4.1, (similar to [56], but here all the nodes need not know about the position of every other node in the network). Nodes in region 1 have the highest probability to forward the traffic, while nodes in region 2 have a lower probability. If no nodes are currently available in region 1, the transmission area is expanded to include region 2, after a timeout. This strategy imposes a priority on neighboring nodes in such a way that, nodes more in line with the direction of the destination have higher probability to forward the message, thereby reducing the delay that traffic suffers on its way towards the destination. This priority is also closely tied to the energy-level of the intermediary node to maximize network lifetime. Consider two nodes, similar with respect to their position from S and D; the node with higher energy will have the higher probability to forward the message towards D.

The nodes that receive the message sent by S calculate their probability of forwarding and based on this information, they either listen or forward the message. Furthermore, upon hearing a transmission within the zone, the remaining eligible nodes drop the message. As the message progresses toward its destination, the node responsible for forwarding the message calculates a new cone and re-iterates the process. This forwarding mechanism is detailed in Algorithm 9.

An important aspect of Algorithm 9 is to calculate the probability used by the nodes to decide whether to forward the message or not. P_n is the probability of forwarding (for an intermediary node, N) and is dependent on α_n (the angle N makes with S, as shown in Figure 4.1) and d_n (distance of N from S). Let the angle of the truncated cone be α_c.

Algorithm 9 Forwarding Messages

```
FORWARD-MSG($\alpha_n$, $d_n$, $\tau(n)$, $M(D)$)
Input: $\alpha_n$, $d_n$, $\tau(n)$, $M(D)$
Output: Result
 1: Calculate $P_n$ using $\alpha_n$, $d_n$, $\tau(n)$
 2: while ! success do
 3:   Generate a random number $P \in [0, 1]$
 4:   if $0 \le P \le P_n$ then
 5:     Calculate $V_L(D)$, the expected location of $D$
 6:     Send limited-directed broadcast of $[V_L(D), M(D), L(N)]$, where $M(D)$ is the message and $L(N)$ is the location of this node $N$
 7:     success = $true$
 8:   else
 9:     Wait for the next time slot
10:     if Msg-Sent by another node before timeout then
11:       Drop request
12:       return Success
13:     else
14:       continue
15:     end if
16:   end if
17: end while
18: return Success
```

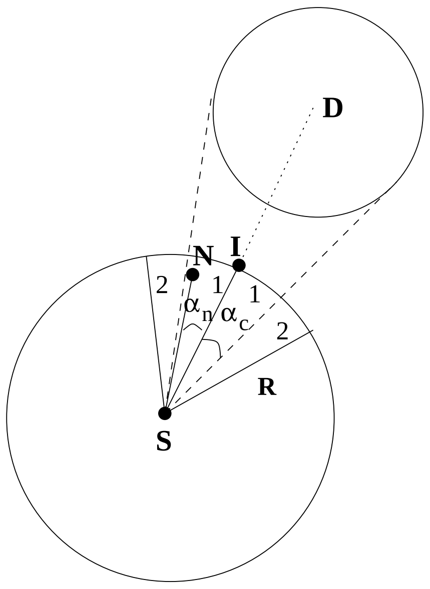


Figure 4.1: Directional Routing

Based on Figure 4.1, it is clear that, if all nodes had equal energy reserves, node I is the best node in zone 1 to forward the message and hence must have the highest probability. We must choose a node within the cone that is the farthest away from the source and is also in the direction of the destination. Let $\tau(n)$ be the residual power at the intermediary node. Furthermore, the probability function must also provide flexibility to balance each factor in the formula. Let R be the transmission range of S. The formula for calculating P_n is given by:

$$P_n = w_1 * \frac{d_n}{R} + w_2 * \frac{(\alpha_c - \alpha_n)}{\alpha_c} + w_3 * \tau(n) \tag{4.1}$$

$w_1,\ w_2$ and w_3 are weights such that, $w_1\ +\ w_2\ +\ w_3\ \leq\ 1$; $P_n = 0,\ d > R\ or\ \alpha_n > \alpha_c$. If all nodes have similar residual power, notice that the value of P_n is highest for a node I (as illustrated in Figure 4.1) whose $\alpha_I = 0$ and $d_I\ =\ R$.

To calculate P_n, an intermediary node needs to calculate α_n. An intermediary node, N only needs to know its relative position with respect to the source and hence, does not require the use of GPS [13]. Consider Figure 4.2; for node N to calculate the value of α_n, it needs to know d_n, R and x. Using a local co-ordinate system such as the one in [58], node S

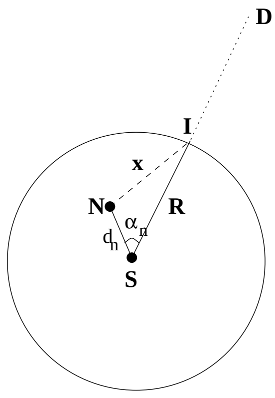


Figure 4.2: Calculation of α_n

calculates the co-ordinates of I (I is the point on the edge of the broadcast radius of S in the direction of D). This information can be embedded in the message to be forwarded. Node N can now calculate its position in the local co-ordinate system of S. Using this information, N calculates d_n, R and x. Now, using the cosine formula, we can derive the value of α_n:

$$\alpha_n = acos\left(\frac{d_n^2 + R^2 - x^2}{2 * R * d_n}\right) \tag{4.2}$$

Another important piece of the algorithm is the *directional routing* that requires each node along the path to re-calculate the *cone* used to forward the message to the destination (shown in Figure 4.3 (part A)). Consider the timeline in Figure 4.3 (part A). At time T_0, node D is at position D_0, at time T, the node is at position D and at time $T + \triangle$, node D is at position D_1. Now, consider the situation, when source S wants to send a message to D. It calculates with a certain probability [2], a region where node D can reside. Node S now calculates the angle α and hence derives the *cone* and sends the message towards the destination. Node S_1 upon receipt of this message, calculates the angle δ and re-calculates a new cone (based on the probability of the new position of D) and sends the message.

Consider part B of Figure 4.3, this shows the movement of the destination from position D to D_1. Assume the expected direction of travel to be β with respect to the $x - axis$ and the expected speed of travel to be v. Let D be the point (x_1, y_1) and D_1 be the point (x_2, y_2)

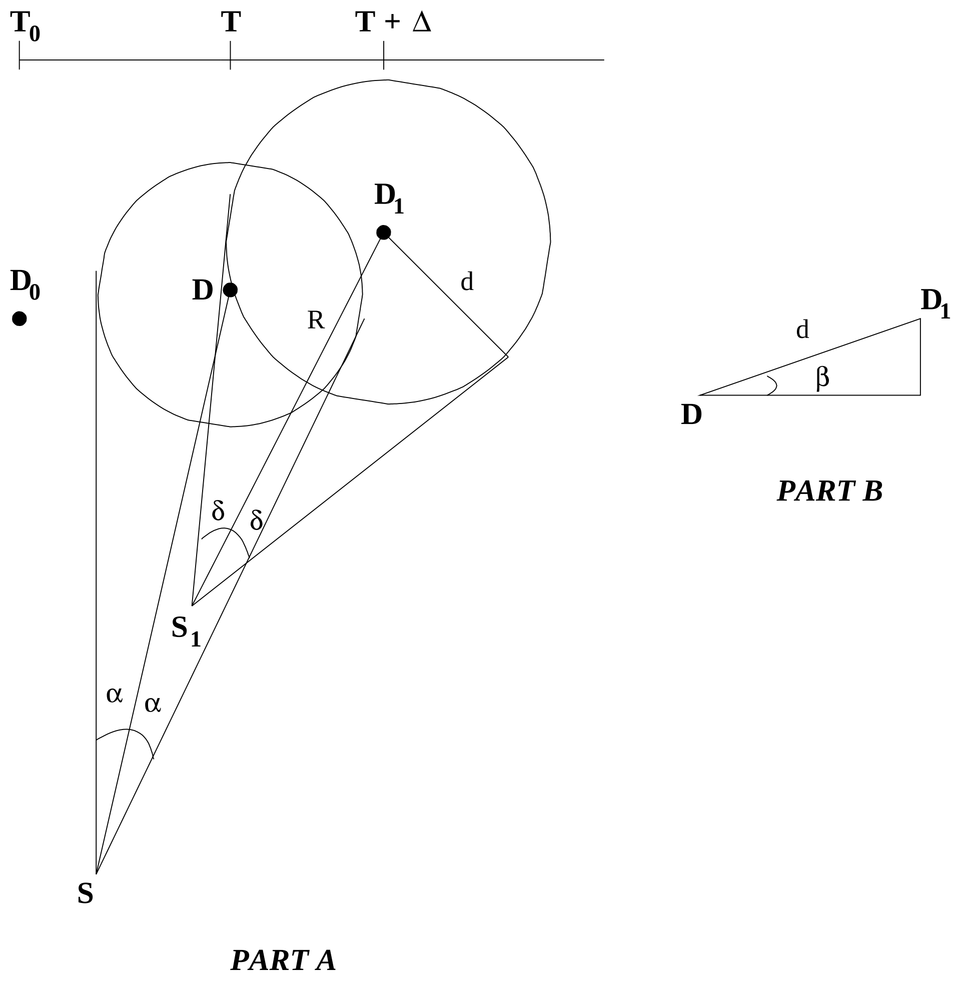


Figure 4.3: Directional Routing as destination moves

in the Cartesian co-ordinate system. We know the position D and we need to find out the new location D_1 in terms of D. This is accomplished using the following equations:

$$d = v * (T - (T + \triangle)) \Longrightarrow d = v * \triangle \tag{4.3}$$

$$x_2 = x_1 + d * cos\beta \tag{4.4}$$

$$y_2 = y_1 + d * sin\beta \tag{4.5}$$

Consider the scenario when node S_1 receives a message from S that is intended for the destination. S_1 needs to calculate the angle for the cone. Now that we have the position D_1 and the value for d, we need to calculate δ based on these values. Let S_1 be the point (sx_1, sy_1). The following sets of equations help us derive δ:

Using the values of x_2 and y_2 from equations 4.4 and 4.5 respectively, we get:

$$R = \sqrt{(x_2 - sx_1)^2 + (y_2 - sy_1)^2} \tag{4.6}$$

Using the value of R from equation 4.6 and the value of d from equation 4.3, we get:

$$\delta = asin\left(\tfrac{d}{R}\right) \tag{4.7}$$

4.2 POWER-AWARE FORWARDING

As discussed in section 4.1.1, the probability P_n, of a node to forward a message to the destination depends both on the distance from the source node and the angle of deviation from the center line of the cone. However, P_n defines only the initial forwarding probability at the instant when a message arrives at the relay node. If the message has not been forwarded the probability of forwarding the message should increase as time elapses. The forwarding probability has to reach 1 by S time slots, where S is a design parameter.

For choosing the next relay node to forward the message to the destination, the best candidate is the node with the maximum P_n. However, the current energy level of the relay node is also important. Letting nodes deplete their energy and die may cause a network partition. The goal is to construct the best available route from the source to the destination while maximizing network lifetime. From the network lifetime point of view, the low energy nodes are the most important and also the most critical. These nodes have used their energy either because they have a lot of data to send or due to the fact that they are located at the confluence of many routes. Letting these critical nodes deplete their energy may cause a network partition and some sources might be unable to reach other destinations, or at least there is an energy and bandwidth overhead associated with re-routing the messages after discovering a broken link (dead nodes). The bottom line is to construct the most efficient route from the source to the destination while maximizing network lifetime.

As a result, in our design, the current energy level of a node affects the probability of choosing this node as the next relay host. This is done by making the rate of increase of P_n, a function of the node's energy level. The higher the energy level, the faster the probability increase rate and vice versa. For example, consider the case when a message arrives at 2 nodes, node A, a high-energy node and node B that has a lower energy level but a higher initial probability of forwarding (P_n). Though node B has an initial higher probability, as time goes by, the probability of forwarding for A increases at a faster rate; as a result of which node A may forward the request message sooner than B and, consequently, it may be chosen as the next relay node instead of B. The *probability of forwarding* function ($\Gamma(e,t)$), thus depends on the energy e and the time slot t at a node and has the following properties:

$$\Gamma(e,1) = P_n \tag{4.8}$$

$$\Gamma(e+1,t) > \Gamma(e,t) \tag{4.9}$$

$$\Gamma(e,t+1) > \Gamma(e,t) \tag{4.10}$$

$$\Gamma(e,S) = 1 \tag{4.11}$$

Since the forwarding probability has to reach 1 by S slots, we have to derive it as a strictly increasing function that starts from P_n and reaches 1 as $t \to S$. Also, the function must produce a family of curves depending on the power at a node. Equation 4.12 has the requisite property and produces a family of curves that start at 0 and reach 1 depending on the value of e.

$$F(x) = 1 - (1-x)^e, 0 < x \leq 1 \tag{4.12}$$

To illustrate the property of this function, we plot this function for varying values of e, as shown in Figure 4.4.

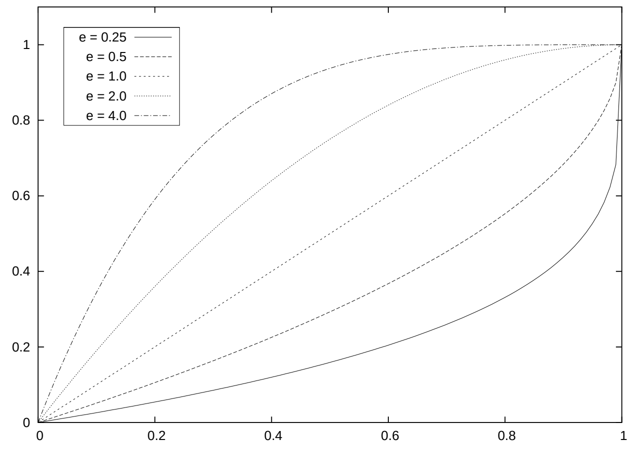


Figure 4.4: F(x) with variable e

Equation 4.12 however, does not satisfy constraint 4.8. Hence, we add the following constraints and substitute $x = \frac{t}{S}$ (t = time slot and S = max time slot).

$$F\left(\frac{t}{S}\right) = \begin{cases} P_n & t = 1 \\ 1 & t = S \end{cases} \tag{4.13}$$

While obeying the constraints in equation 4.13, we can solve Equation 4.12 to get the overall probability of forwarding $\Gamma(e,t)$, which is given in equation 4.14.

$$\Gamma(e,t) \ = \ 1 - \left(\frac{S \ * \ (1-P_n)^{\frac{1}{e}} * (\frac{t}{S} - 1)}{(1-S)}\right)^e \tag{4.14}$$

where P_n = initial forwarding probability as defined by equation 4.1, t = elapsed time slots since the instant of message arrival. Using equation 4.14, we can plot $\Gamma(e,t)$ as shown in Figure 4.5 by varying the value of e from 0.25 to 4 and setting P_n to 0.2. We can observe from Figure 4.5 that equation 4.14 observes the properties stated in equations 4.8 - 4.11. It also has the property that the probability increases faster for a higher value of v and vice versa.

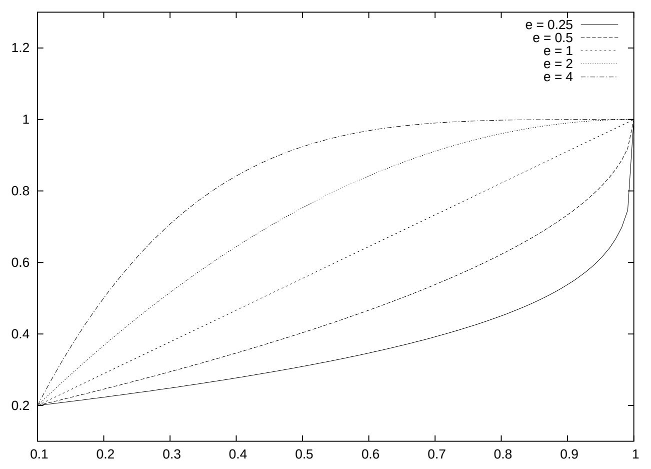


Figure 4.5: Probability of forwarding while changing the value of e

The parameter e in equation 4.14 is a function of the current energy level of the node. When the node has a low energy level, e should be low, which forces the forwarding probability to increase at a slow rate. Also, when the node has a high energy level, e must be high, thus forcing the forwarding probability to increase at a higher rate. It should be noted that, for mid-range energy level the increase of the forwarding probability is almost linear with time. The parameter e can be computed from the node's current relative energy level as follows:

$$e_n = 2^{4\cdot(E_n - 0.5)} \tag{4.15}$$

where E_n is the relative energy level of node n.

It should be mentioned that, although equation 4.14 is a computationally expensive function to evaluate, the wireless node does not need to compute its value online. A matrix $\pi(K, S)$, which defines the function value at each S and for K different energy levels and different probabilities can be computed offline and then used by the wireless node. A node chooses to use the table whose probability is closest to its own P_n. The values of both K and S can be determined at design time. Table 4.1 presents an instance of this matrix when we have $K = 5$, $S = 5$ and $P_n = 0.2$.

Using the knowledge of the energy at a node and the matrix in table 4.1, Algorithm 9 can now be re-written as Algorithm 10.

Algorithm 10 Power-Aware Forwarding

```
POWER-FORWARD-MSG($\alpha_n$, $d_n$, $\tau(n)$, $M(D)$)
Input: $\alpha_n$, $d_n$, $\tau(n)$, $M(D)$
Output: Result
1: Calculate $P_n$ using $\alpha_n$, $d_n$, $\tau(n)$
2: while ! success do
3:    Generate a random number $P$ in $[0, 1]$
4:    if $0 \leq P \leq P_n$ then
5:       Forward-msg
6:       success = $true$
7:    else
8:       Wait for the next time slot
9:       Update $P_n$ for that time slot according to the current residual power of the node
         (table 4.1)
10:      if Msg-Sent by another node before timeout then
11:         Drop request
12:         return Success
13:      else
14:         continue
15:      end if
16:   end if
17: end while
18: return Success
```

Table 4.1: $\Gamma(e,t)$ with $K = 5$, $S = 5$ and $P_n = 0.2$

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
$E_n = 0$	0.2	0.255	0.327	0.434	1
$E_n = 0.25$	0.2	0.307	0.434	0.6	1
$E_n = 0.5$	0.2	0.4	0.6	0.8	1
$E_n = 0.75$	0.2	0.55	0.8	0.95	1
$E_n = 1$	0.2	0.746	0.95	0.996	1

4.3 ANALYSIS OF PILOT

In this section, we analyze PILOT using a probabilistic approach. Consider the total network area to be Λ and the number of network nodes to be $\mathcal{N}$. The distribution of nodes in the network follows a uniform distribution. We would like to calculate the probability of a message reaching the destination. This probability is related directly to the availability of an intermediate node to forward the message towards the destination (the probability is 1 if there does exist an intermediary node). Consider the source S. Let the angle of the truncated code be α and the transmission range be R.

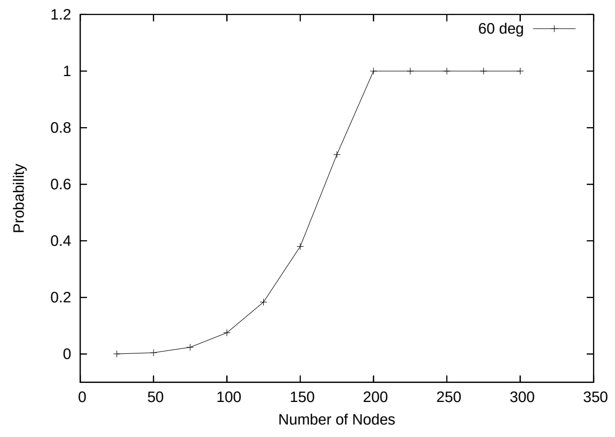
$$Node\ Density(ND) = \tfrac{\mathcal{N}}{\Lambda} \tag{4.16}$$

$$P(intermediary\ node) = min\ \left(\tfrac{\alpha}{2\pi} * \pi R^2 * ND, 1\right) \tag{4.17}$$

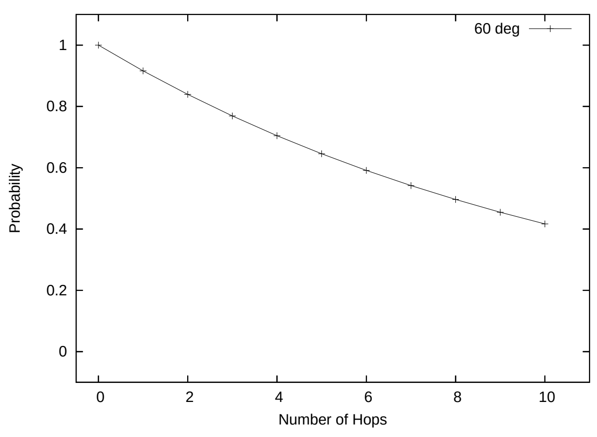
Assuming that there are k intermediary nodes (including the source) and each node i has the angle of the truncated cone as α_i and transmission range R_i, we get the probability of the message reaching the destination, $P(msg)$ as:

$$P(msg) = min\ \left(\prod_{i=1}^{k} \frac{\alpha_i}{2\pi} * \pi {R_i}^2 * ND,\ 1\right) \tag{4.18}$$

To observe the effect that node density and the number of hops have on the the ability of the message to reach the destination, we plot the probability while varying the network density and the number of hops (as shown in Figure 4.6). For Figure 6(a), we assume an average of 4 hops between the source and the destination and vary the number of nodes from 25 - 300. For Figure 6(b), we fix the number of nodes in the network to be 175 and vary the number of hops from 0 to 10. Both the figures are plotted by setting value of the angle of the truncated cone to 60 degrees.



(a) Varying the Network Density



(b) Varying the Number of Hops

Figure 4.6: Probability of Message Reaching the Destination

We can notice from Figure 4.6 that as the network density increases, the probability also increases and finally reaches 1 when the network has enough number of nodes to ensure that the message would reach the destination. Also, in the case when the number of hops to the destination increases, the probability decreases smoothly.

We will now analyze the performance of PILOT by changing the value of the angle of the truncated cone to observe its effect. For this analysis, the angle of the truncated cone is varied using the values from 30, 45, 60 and 90 degrees. To analyze the probability of forwarding with respect to the number of hops, we set the number of nodes in the network to 125. The result of these analysis is provided in Figure 4.7.

Figure 7(a) shows that as the angle of the truncated cone increases, the performance of PILOT becomes better. This is due to the availability of more nodes to forward the traffic towards the destination. Even in the case when the angle of the truncated-cone is

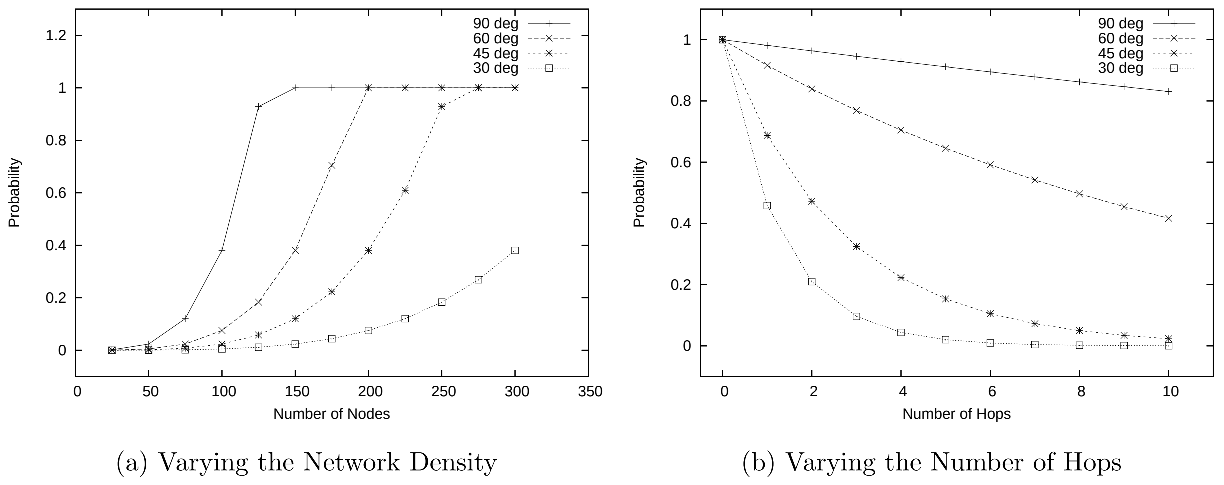


Figure 4.7: Probability of Message Reaching the Destination (Multiple Truncated Cones)

low (45 degrees), the probability of forwarding the message towards the destination reaches 1 as the density of the network grows (> 275 nodes). From Figure 7(b), we can see that the probability of forwarding when the angle of the truncated cone is low falls faster as the number of hops increases. This is due to the lack of nodes available within the truncated cone to forward the message towards the destination.

An important result that we gain from this analysis is the fact that the probability of forwarding in PILOT converges faster when the density of neighboring nodes is high (even when the message needs to traverse a higher number of intermediary nodes). This higher density of nodes available to forward the message can be achieved by using a higher value for the angle of the truncated cone (α_c). During the functioning of PILOT, if there are no nodes available in the primary region (region 1 in Figure 4.1) the transmission area is expanded after a timeout (thus increasing the value of α_c and hence including more neighboring nodes in the decision to forward the message). We can conclude that the message forwarding protocol, PILOT converges, even when the number of available nodes in the network is low.

Given the fact that wireless networks are infrastructureless, the presence of obstacles along the path can lead to network disruption and loss of messages. To observe the effect of obstacles in PILOT, we will now analyze it when there is an obstacle to the message transmission. Let the area blocked (within the truncated cone) due to the transmission hindrance be $\mathcal{B}$. Consider the source S. Let the angle of the truncated code be α and the transmission range be R.

$$Area\ of\ truncated\ cone\ (\mathcal{A}) = \tfrac{\alpha}{2\pi} * \pi R^2 \tag{4.19}$$

$$P(intermediary\ node) = min\ \ ((\mathcal{A} - \mathcal{B}) * ND,\ 1) \tag{4.20}$$

Assuming that there are k intermediary nodes (including the source) and each node i has the angle of the truncated cone as α_i, transmission range R_i and area blocked $\mathcal{B}_i$, we get:

$$Area\ of\ truncated\ cone\ (\mathcal{A}_i) = \tfrac{\alpha_i}{2\pi} * \pi R_i^2 \tag{4.21}$$

$$P(msg) = min\ \left(\textstyle\prod_{i=1}^{k} (\mathcal{A}_i - \mathcal{B}_i) * ND,\ 1\right) \tag{4.22}$$

To analyze the performance of PILOT in the presence of network obstacles, we analyze PILOT with respect to a small obstacle (Figure 4.8) as well as a large obstacle (Figure 4.9). An average of 4 intermediary hops between the source and the destination is assumed while the number of nodes is varied from 25 - 300 in steps of 25. For the analysis with a small obstacle, the obstacle is assumed to be of size 20 degrees (with respect to the angle of the truncated cone). To observe the effect of this obstacle, the size of the truncated cone is varied by using the values 30, 45, 60 and 90 degrees. For the analysis with a large obstacle, the size of the obstacle is assumed to be 60 degrees. In this case, the value of the size of the truncated cone is varied by using the values: 90, 120, 150 and 180 degrees. In both analyses, the probability of forwarding is measured with respect to the network density.

Figure 4.8 shows that as the network density increases, the performance of PILOT increases and it reaches 1 when the angle of the truncated cone is either 60 or 90 degrees. The performance of PILOT is poor when the angle is either 30 or 45 degrees since area formed by the truncated cone is small and the obstacle takes up almost 50% of the area within the truncated cone. This significantly reduces the chances of finding a node to forward the message to the destination since the available area within the truncated cone is so little. Since the probability considerably increases as the angle of the truncated cone increases, over time

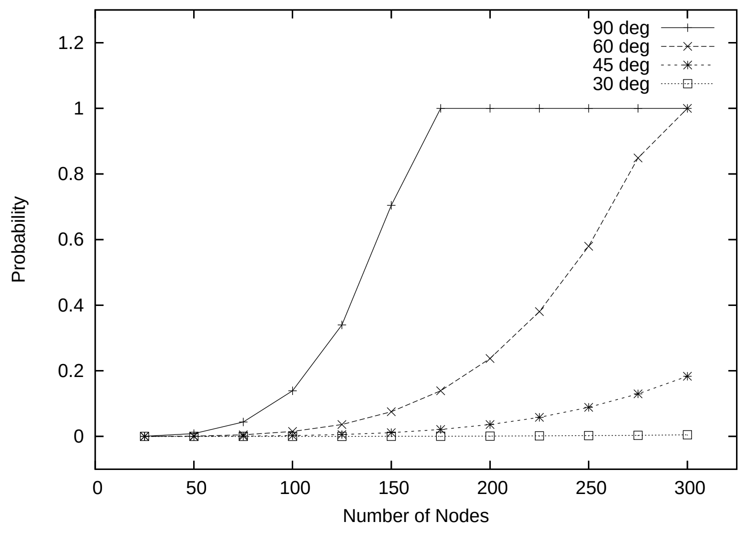


Figure 4.8: Probability of Message Reaching the Destination (Small Obstacle)

a node will be available to circumvent the obstacle and forward the message towards the destination. This follows directly from the architecture of PILOT, where the angle of the truncated cone is increased (after a timeout) if no nodes are available in region 1 to forward the message.

Figure 4.9 shows that the probability of forwarding increases and reaches 1 as the network density increases in all cases, except when the angle of the truncated cone is 90 degrees. The reason for this is that the obstacle takes up 66% of the area within the truncated cone. When the area within the cone is increased by increasing the angle to 120, the probability of forwarding shows a marked improvement and reaches 1 as the density of the network increases. This increase in the angle of the truncated cone helps even in cases when the density of the nodes is not very high ($<$ 150 nodes in the network).

From the analysis of PILOT while using obstacles, we show that the probability of forwarding of PILOT does converge as the density of neighboring nodes in the network increases. In the case, when the obstacle occupies most of the area within the truncated cone, an increase in the angle of the truncated cone results in a higher probability of the message reaching the destination. This increase in the angle of the truncated cone is automatically

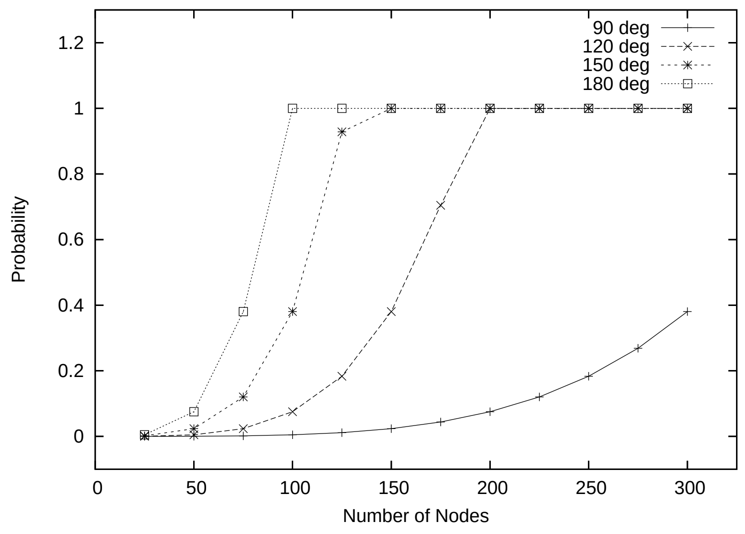


Figure 4.9: Probability of Message Reaching the Destination (Large Obstacle)

achieved by PILOT, when it detects that there are no available nodes in the primary region.

4.4 SUMMARY

This chapter presented in detail the data forwarding algorithm for SARA that forwards traffic in a power-aware location-directed manner. An overview of PILOT was provided in Section 4.1, while Section 4.2 detailed the protocols used to make the traffic forwarding algorithm power-aware by tying the probability of forwarding to the residual power. Section 4.3 concluded this chapter by providing a probabilistic analysis of PILOT.

To limit flooding in the network, PILOT uses the knowledge about the location of the source and the direction of the destination to forward traffic in a truncated cone-shaped manner towards the destination. The intermediary node to forward traffic is chosen by using a priority-based scheme that imposes a priority on the neighboring nodes in a way, such that nodes which are more in line with the direction of the destination have higher probability to forward the message. This reduces the delay that traffic suffers on its way towards the destination. This priority is also closely tied to the residual energy-level of the intermediary node to maximize network lifetime. A probabilistic analysis of PILOT showed that PILOT converged, even when the network density is quite low. When the density of nodes in the network is low, the probability of finding a neighboring node to forward the traffic towards the destination can be increased by increasing the size of the truncated cone. The analysis was also carried out for a network in the presence of obstacles and it also showed that PILOT converged. Due to the presence of network obstacles, however, the average size of the truncated cone needed to be much higher to ensure the presence of neighboring nodes to forward the traffic towards the destination, as compared to the previous case. To further evaluate the performance of PILOT, a simulation analysis comparing PILOT to GPSR [6], AODV [46] and LAR [35] under different network scenarios is provided in Chapter 7.

5.0 USER PRIVACY IN SARA

In this chapter, we present the different schemes that work towards ensuring user privacy in the proposed service architecture. To ensure user privacy in an ubiquitous computing environment, it is imperative to protect the location of the user. The schemes developed to ensure user privacy must also take into account the node mobility and the resource constrained environment. This chapter is divided into four sections: Section 5.1 provides an overview of the security concerns and details related work in this area while Sections 5.2 - 5.4 detail the proposed security mechanisms. Section 5.2 details the Multiple Location Vector scheme, while Section 5.3 details the Node-Proxy based scheme and Section 5.4 details the Random-Proxy based scheme. Section 5.5 provides a complexity analysis of each of the proposed schemes. All these schemes work on the assumption that each node contains a public/private key-pair. For the purpose of generating public/private key-pairs and key pre-distribution, any of the schemes described in [38, 8, 22, 21] can be used. It is also assumed that nodes that are a part of the virtual registry perform their required tasks correctly while replying to discovery requests.

5.1 OVERVIEW

In an ubiquitous computing environment, users often interact directly with the environment through portable devices that they carry. As the users move around, they can still keep in touch and interact with the ubiquitous environment through the use of their portable devices. The information exchanged between the user and the environment often consists of user identity and location information. It is important to be able to protect a user's identity and location information to ensure user privacy. The lack of user privacy may deter users from using the ubiquitous computing environment.

The common method used to protect a user's identity and location information is to encrypt the content information and the traffic to and from the user's portable device. Given the resource constrained environment, this scheme may prove to be infeasible. Another technique is to remove all references to the user's identity while communicating with the ubiquitous environment. It is however, much more difficult to "hide" the location information of the user as well as the source and destination addresses. One scheme to hide the addresses of the source and destination is to use broadcast messaging. This is not a feasible solution due to the large number of messages generated in the network and the consequent communication cost. Another scheme to protect node addresses is through the process of anonymization. The simplest anonymization technique is to use a "proxy" node to relay the traffic to/from the intended recipient. To increase the security of the scheme, multiple "proxy" nodes could be used to forward traffic in the network.

There are several recent projects that protect user identity through anonymity in ubiquitous environments [20, 32]. These schemes extend the rules and practices established in Internet Privacy [37, 29] (which focus on obscuring a user's IP address) to allow for them to be used in ubiquitous environments. [32] provides a scheme for anonymizing user's communication in ubiquitous environments by using a hierarchy of "Mist Routers" that preserve privacy and hide information about the source and destination.

Though proxy based schemes provide a solution to protect user identity, they are still vulnerable to traffic analysis and hence are not suited for protecting the location information of users. To overcome the limitations of proxy based solutions, we create multiple "dummy" user locations in the network and use a routing scheme similar to [10]. Messages are forwarded to a particular location, rather than a node. To forward a message, multiple messages are created such that they appear to originate from the different user locations and these messages follow a sequence of locations (similar to Landmark Routing [45, 44]) consisting of different landmark points before arriving at the destination location. Figure 5.1 has an example of this scheme where there are 5 different routes between node N and node C, with each route visiting different landmark points (L_1 to $L11$). A node forwarding the message only knows the *next* location, but does not know which location the message originated from. This kind of routing is also referred to as *onion routing*, where each hop *peels* a layer of the route to find out the next hop. Information about the destination and the packet payload are encrypted using the public-key of the recipient node. The destination node decrypts the message using its private-key to retrieve the information. Algorithm 11 details the mechanism by which a node N responds to a node C (using the location vector of C) by building K-different onion routes.

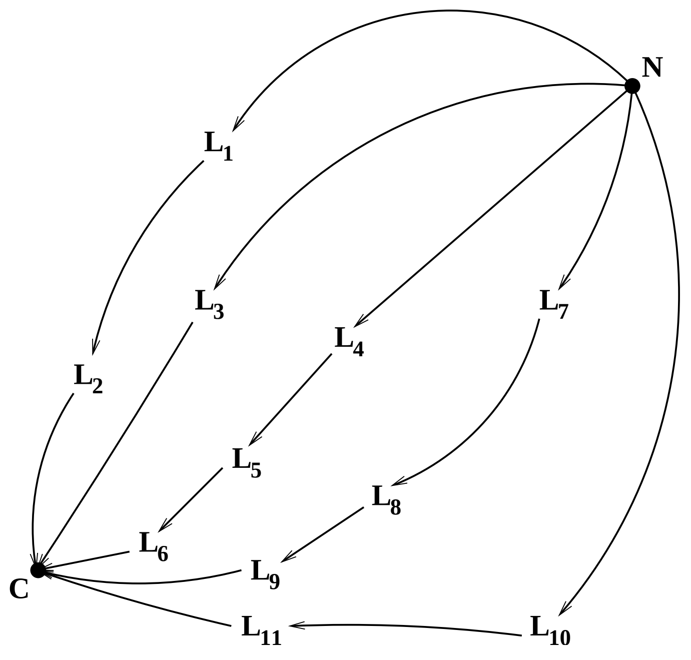


Figure 5.1: Onion Routing using Landmarks

5.2 MULTIPLE LOCATION VECTOR

In this section, we detail the *Multiple Location Vector* scheme, wherein the node registers multiple *location vectors* with the *virtual registry*. This allows the mobile node to "mask" its current location by using multiple locations in the network.

While registering its location vector with the virtual registry, a node N registers upto K different location vectors. It is the responsibility of node N to pre-calculate the other $K-1$

Algorithm 11 Multiple Replies

```
ONION-REPLY($K$, $LV_C$)
Input: $K$, $LV_C$
Output: Result
1: Build onion routing path for each of the $K$ paths that consists of a set of landmark
   points to traverse before reaching the location of $C$
2: Send the reply along each of the $K$ paths
3: return
```

locations and make sure that those locations are plausible. It should also not be possible to predict the correct location vector from the K registered location vectors. Algorithm 12 details the steps by which a node N registers K different location vectors by making use of Algorithm 3.

Algorithm 12 Register K-different LVs

```
REG-MULT-LVS($H$, $K$, $LV_N$)
Input: $K$, $LV_N$
Output: Result
1: Generate $K$-different location vectors ($LV_N^i$ $i = 1..K$) using $LV_N$
2: for i = 1 ... $K$ do
3:    LV-REGISTER($N_{id}$, $H$, $LV_N^i$)
4: end for
5: return
```

When a query for the location vector of N is received by the virtual registry, the registry replies with the K-different location vectors of N. A node C that wishes to establish a connection with N must now send its request to all K locations. Node N upon receipt of the query uses Algorithm 11 to reply to C. Thus, the location of N is hidden from node C since C does not know which of the K locations of N is the correct one.

This scheme, however, has a drawback since node C can launch a timing attack. Consider the scenario when node C sends requests to each of the K locations of node N one after another, rather than simultaneously. C waits for an amount of time after sending out a request. If it receives a response from sending the message to that location, it can infer the correct location of N and hence its location vector. This attack can however be mitigated, if the source node N chooses a random delay before replying to C.

Lemma 5.1. *In the absence of the timing attack, the Multiple Location Vector scheme ensures the user privacy of a node in the event of an "attacker" compromising a set of nodes in the network (except the source and destination).*

Proof: By compromising nodes in the network, an attacker can gain access to traffic between the source and the destination node. The information gained by the attacker contains the destination location information, but this location information contains multiple locations where the destination can potentially exist. Even if the attacker captures a node along each of the multiple different paths, the attacker has no method of verifying whether the destination does indeed lie along that path. The location of the source is protected in a similar manner. The correctness of this scheme follows from the fact that even though an attacker has the location information of the destination, he/she does not know the exact location and hence cannot predict the location of the user.

5.3 NODE-PROXY BASED

In this section, we detail the *Node-Proxy Based* security scheme used in SARA to ensure user privacy. A node using this scheme registers the *location vectors* of its proxies, rather than its own location vector. These proxies are recruited by the node by using Algorithm 13. A node's location vector cannot be associated with a particular node since the ID of the node is associated with the location vector of a proxy, thus preserving node anonymity.

Before registering its location vector, a node N needs to recruit its proxies. Node N broadcasts a message ($PROXY\_RECRUIT$) with its *unique id* (ID_N) to recruit proxies. Nodes that choose to act as a proxy on behalf of N reply to the request by providing their location information. A proxy node (NP) can choose to provide K-different location vectors. There is no way to associate a location with a particular node, since the *ID* of the proxy node is not available. Upon receiving replies from the proxy nodes, node N sends its location vector to the proxies. Since a node can serve as a proxy for more than one node, there is a need to index the location vector information. Since each node in the network has a unique id, the id of the node is used as an index for the location vector information. Algorithm 13 details the mechanism by which a node N recruits proxies, while Algorithm 14 details the steps involved when a node accepts the proxy request and becomes a proxy (ID_N is the id of node N).

Algorithm 13 Recruit Node Proxy

```
RECRUIT-PROXY()
Input: Void
Output: Result
1: Broadcast request $PROXY\_RECRUIT$ along with $ID_N$ and $LV_N$
2: if ((n = recv_reply()) $\leq$ 1) then
3:     REG-MULT-LVS($H$, $K$, $LV_N$)
4: else
5:     REG-MULT-LVS($H$, $K$, $LV_{NP}$)
6: end if
7: return
```

When a query for the location vector of N is received by the virtual registry, the registry replies with the location vector of the one of the proxies of N (NP_1). A node C that wishes to establish a connection with N must now send its request to all K locations of NP_1. Node C now constructs a message that contains its request (encrypted using the public key of N) and the public key of N and forwards this message to NP_1. The id of N (ID_N) is used by NP_1 as an index to retrieve the location vector of N and the message is forwarded to N. Algorithm 15 details the steps involved when a node C tries to contact a node N (It is assumed that node C has queried for LV_N and has received the location vector (LV_{NP_1}) of node proxy NP_1, PK_N is the public-key of node N).

Algorithm 14 Accept Node Proxy

```
ACCEPT-PROXY()
Input: Void
Output: Result
1: Recv $PROXY\_REQUEST$
2: if accept then
3:     Store $ID_N$ and $LV_N$
4:     Reply back to $N$ with $msg$ $PROXY\_ACCEPT$ along with $LV_{NP}$
5: end if
6: return
```

Algorithm 15 Contact Node (Node Proxy based scheme)

```
CONTACT-NODE-NP($LV_{NP_1}$)
Input: $LV_{NP_1}$
Output: Result
1: Construct a $msg$ $CONTACT\_NODE$ containing the request (encrypted using $PK_N$) and
   $ID_N$
2: Forward $msg$ to $NP_1$
3: $NP_1$ receives the $msg$
4: Lookup ($ID_N$) to get $LV_N$
5: Forward $msg$ to $N$
6: return
```

The Node-Proxy based security scheme has several advantages. During subsequent queries for the location vector of node N, the registry can randomly select any of the available proxies for N. The location information of the proxy is hidden from the node and vice-versa. Node C cannot collude with the nodes from the virtual registry since the registry only contains the location information of the proxy nodes. Node C and the proxy nodes cannot collude with each other since the identity of the proxy is not known to node C and vice-versa. Also, the timing attack (mentioned in Section 5.2) does not work, since the message to N goes through a proxy.

The Node-Proxy based security scheme does have a few disadvantages though. In the event that a proxy node changes its location vector, the proxy would need to contact all the nodes for which it acts as a proxy and update this information. These nodes must in turn update this change in the location vector at their respective virtual registries. In case a node is not able to recruit proxies due to a lack of neighboring nodes or due to the unwillingness of neighboring nodes to serve as proxies, this scheme degenerates to the Multiple-Location Vector scheme. Also, the location information of the node N is available to the proxy node and a proxy can perform a timing attack to locate N. One method to mitigate this attack is through the use of trusted proxies. If the network does not provide the availability of trusted proxies, the timing attack from the proxy node can be mitigated through the use of multiple proxies. Consider the scenario when node N recruits multiple proxies (NP_1, NP_2, NP_3). Node N could provide NP_1 with the location vector of NP_2, while providing NP_2 with the location vector of NP_3 and providing LV_N to node NP_3 ($C \rightarrow NP_1 \rightarrow NP_2 \rightarrow NP_3 \rightarrow N$). This increases the security of the scheme, since the proxy nodes only contain the location information about the next node and they do not know whether the next "hop" node is the destination node (N) or not. In this manner, the location information of node N is hidden from the proxies.

Lemma 5.2. *The Node-Proxy Based scheme ensures the user privacy of a node in the event of an "attacker" compromising a set of nodes in the network (except the source and destination).*

Proof: By compromising nodes in the network, an attacker can gain access to traffic between the source and the proxy or between the proxy and the destination. If the attacker succeeds in capturing an intermediary node between the source and the proxy or the proxy and the destination, the information gained contains the location information to send the message to. The attacker does not know if the location information belongs to the destination or the proxy. Also, each message can be sent along K different paths (due to the proxies and the destination registering upto K different location vectors). In case the attacker compromises a proxy node, he/she has access to the identity of the destination and its location information. The attacker can then choose to perform the timing attack. This scheme can be mitigated by using multiple proxy indirections. In each of the above cases, the attacker either knows the id (in the case of the destination) or the approximate location information. The correctness of the scheme follows from the fact that the attacker cannot locate a user since it has either the id or the location information, but not both.

5.4 RANDOM-PROXY BASED

This section details the *Random-Proxy Based* security scheme used in SARA to ensure user privacy. In this scheme, the *virtual registry* replying to a query for a node's location vector constructs a "path" of nodes to traverse before reaching the destination node. The location information of each node along the path is encrypted by using the public-key of the preceding node. Similar to the Multiple Location Vector scheme (Section 5.2), a node N registers upto K different location vectors by using Algorithm 12. The difference between the Random-Proxy based scheme and the Multiple Location Vector scheme lies in the discovery phase.

Consider the scenario when a node C tries to locate a node N and sends a query to the virtual registry containing the location vector of N. The nodes in the registry reply with a *path* to N, where the *path* is a list of randomly chosen proxy nodes (each of which have registered their location vector at this registry). These proxy nodes act as a redirection and each one of these proxies can have upto K different location vectors (as per their registrations). Each node belonging to the list (other than the destination) is referred to as a random proxy (RP). This proxy list is encrypted (using the public-key of the proxies) in a way such that a node in the list only knows the location of the next hop. This entire message is encrypted using the public-key of C. Upon receipt of the reply from the virtual registry, node C decrypts the message using its private-key to get the location vector of the first proxy (LV_{RP_1}). Node C now constructs a message that contains its request (encrypted using the public key of N), the public key of N and the proxy list and forwards this message to RP_1. Node RP_1 decrypts the proxy list to get LV_{RP_2} and sends the message to RP_2. Continuing in this manner, the message finally reaches the destination node N.

Consider an example where the path from C to N uses random proxies RP_1 and RP_2 such that: $C \rightarrow RP_1 \rightarrow RP_2 \rightarrow N$. In the given proxy list, the list is encrypted using: $E_{PK_C}\left(LV_{RP_1}|E_{PK_{RP_1}}\left(LV_{RP_2}|E_{PK_{RP_2}}\left(LV_N\right)\right)\right)$. To forward the message (*msg*) C decrypts the *proxy list* to get LV_{RP_1} and forwards the *msg* to RP_1; RP_1 then decrypts the *proxy list* to get LV_{RP_2} and forwards the msg to RP_2; RP_2 decrypts the *proxy list* to get LV_N and forwards the *msg* to N. Algorithm 16 details the steps involved when a node C attempts to contact another node N.

Algorithm 16 Contact Node (Random Proxy based scheme)

```
NODE-CONTACT-RP()
Input: Void
Output: Result
1: $VR$ receives the $LV\_DISCOVER$ request from $C$
2: Construct the path consisting of a set of randomly selected "proxy" nodes from among
   those nodes that have registered their $LV$s with this registry
3: Encrypt the $LV$ of each proxy with the preceding proxy's public-key
4: Encrypt the proxy list with the public-key of $C$
5: $C$ constructs a msg $CONTACT\_NODE$ containing the request (encrypted using $PK_N$),
   proxy list and $ID_N$
6: $C$ decrypts the proxy list and forwards the msg to the location of the first proxy
7: Continuing in this manner, the message finally reaches $N$
```

The Random-Proxy based scheme has several advantages. During subsequent queries for the location vector of node N, the registry can alter the sequence of the proxies in the proxy list or chose a different set of proxy nodes. The location of the proxies are hidden from each other and also from the source and destination nodes. Node C and the proxy nodes cannot collude since the identity of the proxy is not known to node C and vice-versa. Also, the timing attack (mentioned in Section 5.2) does not work, since the messages from C to N must go through a proxy. Each of the random proxy nodes can choose to register upto K-different location vectors, thus enhancing the security of this scheme.

This security scheme does have a few disadvantages though. Due to the fact that the proxy list is generated using random proxy nodes that have registered their location vectors at this registry, it may so happen that there may not be enough proxy nodes available (due to the fact that not many nodes may have "hashed" to this registry). In the scenario that there are no nodes available to act as proxies, this scheme degenerates to the Multiple-Location Vector scheme. If node C and the nodes belonging to the registry (storing LV_N) collude together, the registry nodes could just send the location vector of LV_N directly to C. This enables C to launch a timing attack (mentioned in Section 5.2). Also, the indirection due to the proxies will lead to delays in traffic reaching the destination.

Lemma 5.3. *In the event of an "attacker" compromising a set of nodes in the network (except the source and the destination), the Random-Proxy Based scheme ensures the user privacy of a node is preserved.*

Proof: information that an attacker can gather by compromising nodes in the network is the traffic between source and the proxy or traffic between proxy and destination or traffic between proxies. In each of the aforementioned scenarios, the information contains the ID of the source and destination and the location information of the intermediate proxy nodes. The privacy of the proxy node is preserved since the identity of the proxy is never revealed. The location of the destination is protected from the source since the source only knows the location information of the first proxy node (this is due to the fact that the "path" to the destination consisting of random intermediate proxy nodes is encrypted using the public-keys of the nodes in a manner such that, upon decryption of the message, the corresponding node only knows the location of the next hop). The location of the destination is protected from the proxy node since the proxy node only knows the location information of the next "hop" and it does not know if the next hop node is the destination or not. The source location information is protected in a similar manner. If the attacker succeeds in capturing a proxy node, he/she only knows the location of the next node along the path, but not its identity (since the information is encrypted using public-keys). The correctness of this scheme follows from the fact that a node's id and its location information cannot be verified together.

5.5 COMPLEXITY ANALYSIS

In this section, we compare the three proposed schemes with respect to their communication costs. Three metrics are chosen, namely: set up communication cost (cost to set up the scheme), update communication cost (cost to update the location vector of the node) and the generic communication cost for node interaction. We assume the average path length to be m and the number of average location vector updates to be u.

In the Multiple Location Vector scheme, the set up communication cost is zero since the node follows the normal procedure of registering its location vector and its objects. The only difference is in registering the location vector, where instead of registering one location vector, the node registers upto p different location vectors. The communication cost to update a node's location vector is $O(u * m)$. The generic communication cost is related to the number of different routes followed along with the average path length and is $O(p * m)$.

For the Node-Proxy Based scheme, the set up cost is directly related to the number of neighboring nodes that agree to serve as a proxy. Assuming that the number of proxies is k, the set up cost is $O(k + 1)$ (1 broadcast message along with k replies from the proxies). To update the location vector of a proxy, the proxy needs to contact all the nodes for which it is serving as a proxy and each of those nodes must in turn update this information at the location they registered. The cost associated with this is given to be $O(u * k * m)$. The generic communication cost is just the cost of sending the message from the source to the proxy plus the cost of sending the message from the proxy to the destination and it works out to be $O((k + 1) * p * m)$ (assuming k intermediate proxies).

Similar to the Multiple Location Vector scheme, the set up cost for the Random-Proxy

Based scheme is zero and the communication cost to update a node's location vector is $O(u * m)$. The generic communication cost, however depends on the number of proxies that need to be traversed before reaching the destination. Let the number of intermediary proxies be n. The generic communication cost is then given as $O((n + 1) * p * m)$.

Table 5.1 lists the costs associated with each of the three different schemes. In the table, m represents the average path length, p represents the number of location vectors, k represents the average number of node proxies, u represents the average number of location vector updates per node and n represents the average number of random proxies.

Table 5.1: Comparison of the communication costs

	Multiple Location Vector	Node-Proxy	Random-Proxy
Set up	0	$O(k + 1)$	0
LV Update	$O(u * m)$	$O(u * k * m)$	$O(u * m)$
Node Interaction	$O(p * m)$	$O((k + 1) * p * m)$	$O((n + 1) * p * m)$

From Table 5.1, we can see that the Multiple Location Vector scheme is the most resource conscious since it does not incur a large cost for node interaction. Also, there is no set up cost for this scheme. Both the Random-Proxy Based scheme and the Node-Proxy Based scheme have a much larger communication cost associated with them. The Node-Proxy Based scheme does have a higher cost associated with setting up the scheme (which is still linear with respect to the number of neighboring nodes). The cost of location vector update however, is quite high for the Node-Proxy Based scheme. In a network, where the location vectors of the nodes is mostly accurate and does not change often, this cost becomes negligible.

5.6 SUMMARY

This chapter presented the different schemes developed to ensure user privacy in SARA. Section 5.1 provided an overview of the privacy and security concerns in ubiquitous environments along with the related work in this area. Section 5.2 detailed the Multiple Location Vector scheme, while Section 5.3 detailed the Node-Proxy based scheme. Section 5.4 concluded this chapter by detailing the Random-Proxy based scheme. Section 5.5 provides a complexity analysis of each of the proposed schemes.

In the *Multiple Location Vector* scheme, the node registers multiple *location vectors* with the *virtual registry*. This allows the mobile node to "mask" its current location by using multiple locations in the network. When using the *Node-Proxy Based* scheme, a mobile node registers the *location vector* of its proxies. These proxies are chosen by the node during the bootstrap process. A node's location vector is not associated with a particular node since the node registers the location vector of its proxies during the registration process. In the *Random-Proxy Based* scheme, the *virtual registry* replying to a query for a node's location vector constructs a "path" of nodes to traverse before reaching the destination node. In this scheme, the location information of each node along the path is encrypted by using the public-key of the preceding node. Depending on the network conditions and the resource constraints, any one of the above security schemes may be chosen to ensure user privacy in SARA.

6.0 DESIGN AND IMPLEMENTATION

This chapter details the design and implementation of the proposed service architecture. Architectures and protocols developed for infrastructureless networks are normally tested using simulations but without actual implementation; it is difficult to perceive how efficient and effective the protocol would perform in the real world. Design factors involved in designing a working model of the proposed service architecture are detailed in Section 6.1. Section 6.2 presents the overall design and talks in detail about the different components associated with the implementation. Section 6.3 details the various modules and data structures that are used as part of this implementation along with their interactions. This chapter concludes with Section 6.4, which talks in detail about how the implementation tried to closely follow the software engineering methodologies outlined in Section 6.1.1.

6.1 DESIGN CONSIDERATIONS

6.1.1 Software Design Principles

Any complex implementation is both time consuming and expensive in terms of the resources involved. It is necessary to adhere to some basic software engineering principles so as to minimize the effort required and the risks involved. The basic tenets of a well designed software system are emphasized below.

- *Software Modularity*: As a piece of software grows in size, the complexity and the dependencies between the various parts of the software increases. This will adversely affect the reliability of the software unless the complexity can be reduced. *Modularity* is a software engineering concept by which a complex piece of software is broken down into a number of smaller logically related units. Controlled access to the services (routines, as well as data objects) provided by the module is through a well defined interface, which is exported by the module. The interface, thus exported should be flexible and easy to use. Using modularity, it is possible to develop a complex software system incrementally, by testing each module separately before incorporating them in the main system.

- *Software Reuse*: A well defined modular system allows for re-usability of code in the same module as well as in other modules. Software reuse allows for cutting down on development and testing costs, thus increasing productivity, quality and reliability. Increasing reuse of software results in the software being better tested and debugged, thus leading to an improvement in the quality of the software.
- *Software Portability*: With increasing number of hardware platforms available, it is important for any software implementation to take into account the issue of portability. *Portability* is the ability to take a piece of software written for one platform and make it run on another platform. This is a desirable criteria to have in any software system, because it allows for the software to be written once and used on many different hardware platforms without too many modifications.
- *Software Efficiency*: To achieve maximum efficiency, parts critical to the performance of a software system are written in a way so as to exploit the existing hardware. But, this comes at a cost to portability and becomes a major issue if the majority of the software is system dependent. The easy way around it is to try and confine system dependent code only to the very essential sections and write more portable code on top of it.
- *Software Maintenance*: During the course of building a complex software system, many components are added or dropped. This leads to constant changes in the software, that may or may not be reflective of the original design. As the complexity increases, it becomes more and more difficult to comprehend and keep track of the changes, due to which bugs may be introduced inadvertently. Keeping the software modular reduces the complexity of maintenance, since changes can be incorporated at the module level without having to worry about changes to the complete system. Only the interfaces that the module exports need to be changed, if at all.

The main goal while implementing such a large and complex system was to keep it simple so that future ideas could be incorporated easily. After reading the literature on the implementation of *routed* and *ospfd* [64] for Linux, we decided to implement the system predominantly in user space. Implementation of the system in user space has the advantage of being able to use user level libraries, thus making the task of writing the program easier. It also means that the system would be easily portable to other architectures.

6.1.2 Implementation Platform

The choice of the operating system to implement the protocol would be critical in the long run, as we would be using some of the services offered by it. Also, since we would be working closely with the kernel for some aspects of the implementation, it was important for us to choose the operating system carefully. We used the following guidelines as markers while choosing Linux as our operating system of choice for this implementation.

Linux is emerging as a strong contender for mainstream operating systems. Increasing number of hardware manufactures have begun to recognize the potential of Linux, as a result of which more and more hardwares are now being supported. Over the last few years, Linux has also emerged as a platform of choice for embedded systems. PDAs and smaller devices are being equipped with Linux as the operating system. This means that Linux is and will be a widely supported platform.

In the course of development, we had to interact closely with some aspects of the kernel (like RAW sockets and routing tables), which meant that having access to the source code and good documentation to go along with that to understand the source code was imperative. The Linux kernel is freely available under GNU's Public License (GPL) [19]. It provides extensive documentation and help from other developers. Also, any system developed under the GPL can be distributed freely.

Linux is constantly evolving with more and more features being added. With the utilities and development libraries available, Linux offers a rich platform for software development. The GPL allows us to make the protocol available freely. This would help in improving the protocol as other users can deploy the system, suggest changes and modify the system.

6.2 OVERALL DESIGN

The overall design on which this implementation is based on is shown in Figure 6.1. This design mirrors the logical organization of the service architecture that was presented in Figure 1.1 in Chapter 1.

The highest layer in the framework is the user running applications on their device. The user may choose to perform an object discovery or an object registration by using the interface exported by these modules. To allow for and incorporate mobility in the network, these modules interact with the mobility management module. These three modules use the message forwarding protocol, PILOT, to be able to route traffic to the intended destination. The implementation details for the service architecture containing object registration and discovery services, mobility management and PILOT are further extrapolated in Section 6.3. Once a packet is available to be forwarded, PILOT uses RAW sockets to create its own IP header in the packet before sending the packet directly to the MAC layer, which in turn injects this packet into the network.

The reason for using RAW sockets to build and send packets is the flexibility provided by them, since they allow applications to handle IP packets directly by bypassing upper layer protocols, such as TCP (Figure 6.2). The application can choose to form the IP header and the payload by itself. Using RAW sockets, we can construct our own payload and add protocol headers to the packets before they are sent out. This allows us to create a protocol type that is used by the system (*MSG_PROTO*). Another advantage of using RAW sockets is the fact that the system can be built in user space while the RAW socket functionality can be accessed by using well-defined system calls into the Linux kernel.

6.3 IMPLEMENTATION MODULES

In this section, we detail the implementation modules used and their interaction. The overall design of the system was modularized to allow for ease of implementation and testing. Each main component of the service architecture was implemented as a separate module.

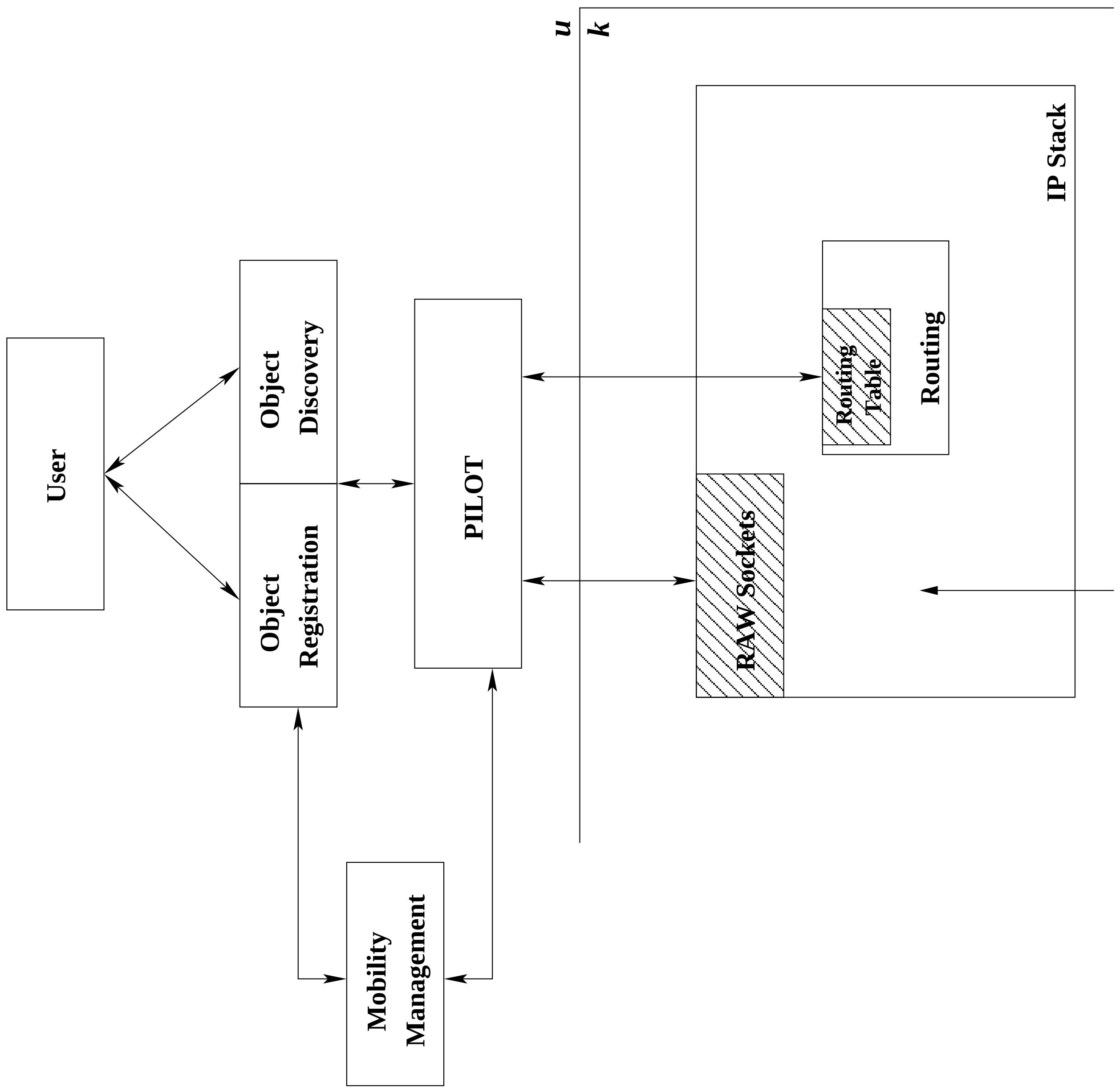


Figure 6.1: Design of the Implementation (u = user level, k = kernel level)

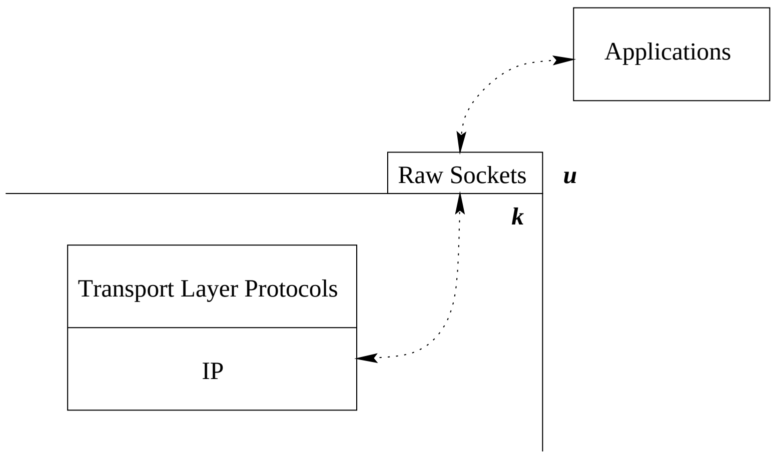


Figure 6.2: RAW Sockets

In the case of the object registration and discovery modules and the daemon, they were implemented as separate processes. Figure 6.3 shows the different modules in the system and their interaction with each other. Sections 6.3.1 - 6.3.4 provide a detailed description about each of the modules.

6.3.1 Shared Memory

This module is central to the system implementation. It is used as an inter-process communicating tool to interact between the registration and discovery modules and the daemon module. The shared memory is created during the bootstrapping phase and is deleted when the program quits. It is modeled as a queue and has the following data structure:

```
typedef struct SERDATA_LIST
{
     int last;
     struct ser_data msg[MAX_VALUE];
}
```

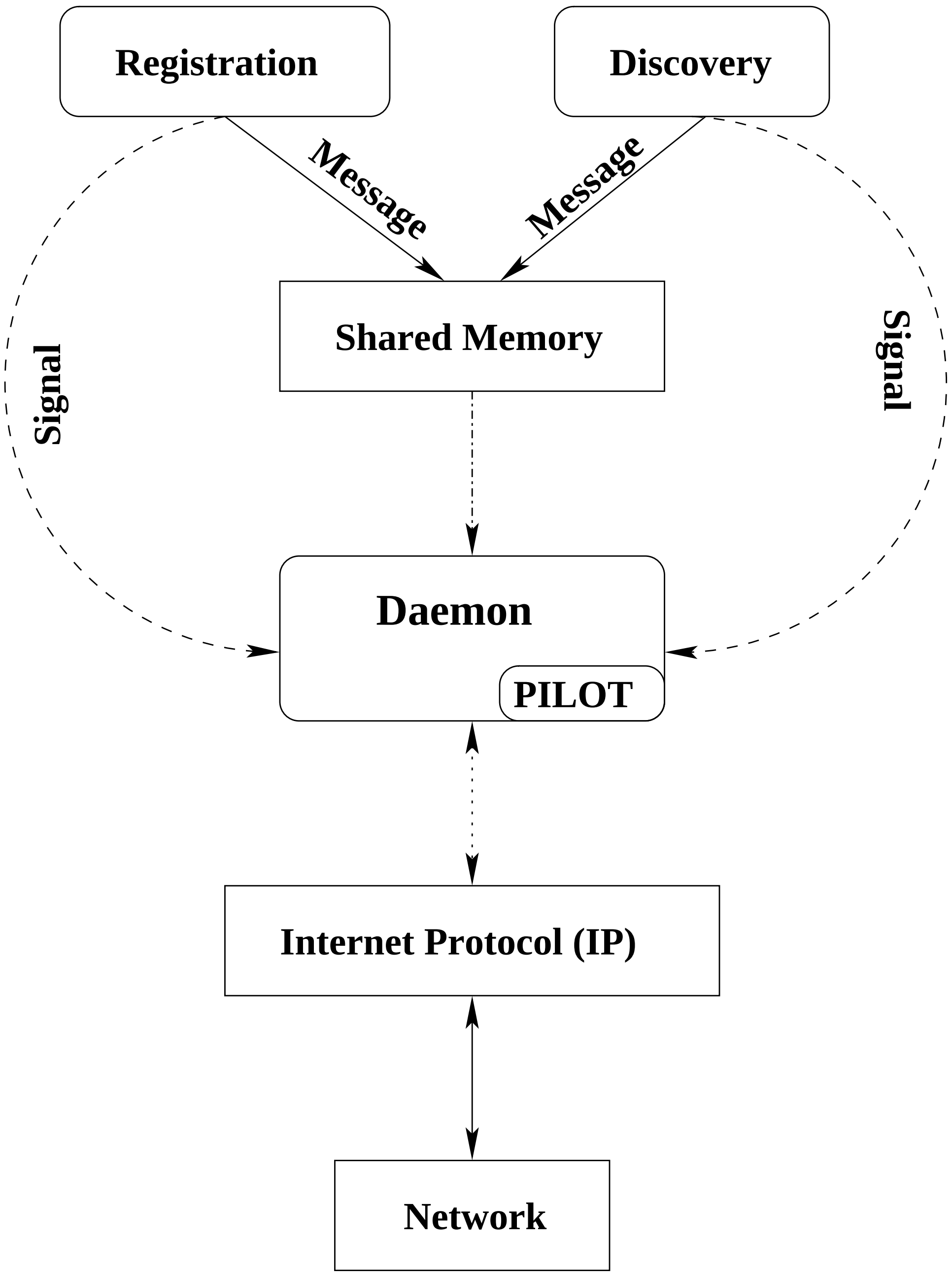


Figure 6.3: Implementation Modules

The shared memory contains a queue of structures, each of which are of type *ser_data* and the size of the queue is set to *MAX_VALUE*. The variable `last` maintains the current offset in the queue, which indicates the location where a new element can be added to the queue. Any time a new element is added to the queue or deleted from the queue, the index `last` is appropriately incremented or decremented. To ensure that the queue is not full, the index `last` is checked every time an element is added to the queue. When the queue is created for the first time, `last` is set to -1. The datatype *ser_data* has the following structure:

```
struct ser_data
{
     MSG_TYPE type; // Type of the message
     int id;        // Entity ID
     int x;         // X co-ordinate of the destination
     int y;         // Y co-ordinate of the destination
}
```

Due to the fact that this module is shared between the other 3 modules, it is imperative to ensure that the data in the queue is accessed and changed by atmost one process at a time. To ensure that only one process accesses the shared memory at anytime, semaphores were used. The major functions used in this module are provided in Table 6.1.

6.3.2 Object Registration

This module is responsible for registering the objects with the network and is invoked in the event when a user wants to register their objects with the network. This module is modeled as a separate process that uses the network-wide hash function to calculate the x and y co-ordinate to send the required information to. This information is inserted into the shared memory (along with the appropriate message type *REGISTER*) using the function *insert_msg()* provided by the shared memory module and a signal is sent to the daemon process. Upon receipt of the signal, the daemon process processes the message from the shared memory and creates an appropriate packet before using PILOT to forward the packet to the destination.

Table 6.1: Important Functions in the Shared Memory module

Function	**Description**
insert_msg()	Insert a message into the shared memory queue.
remove_msg()	Remove a message from the shared memory queue.
print_msg()	Print the existing messages in the shared memory queue.
sem_create()	Create the semaphore.
sem_wait()	Wait on a semaphore.
sem_signal()	Signal the semaphore.
sem_close()	Destroy the semaphore.

6.3.3 Object Discovery

This module is responsible for discovering objects in the network and is invoked in the event when a user wants to discover an object in the network. This module is modeled as a separate process that uses the network-wide hash function to calculate the x and y co-ordinate to send the required information to. This information is inserted into the shared memory (along with the appropriate message type *DISCOVER*) using the function *insert_msg()* provided by the shared memory module and a signal is sent to the daemon process. Upon receipt of the signal, the daemon process processes the message from the shared memory by creating the appropriate packet before using PILOT to forward the packet to the destination. Since the functioning of this module is very similar to the Object Registration module, they share a reasonable amount of code space.

6.3.4 Daemon

This module is the most important process in the system. The Daemon works asynchronously in a distributed fashion as it is always running on every node in the network. During the bootstrapping process, the Daemon process initializes the system by creating and initializing the shared memory, creating the RAW socket for communication and by initializing the other required data structures. Its primary functions are: (i) process the signals received; (ii) process the object registration and discovery requests, and finally (iii) process the packets received. We will look at each one of the functions separately in the following paragraphs.

Upon receipt of a signal, the daemon checks the shared memory for the message. Depending on the type of the message (*REGISTER* or *DISCOVER*), the daemon creates the appropriate IP headers for the packet and constructs the packet before sending the packet out into the network by using PILOT. The format of the packet header used is shown in Figure 6.4, while the format of the packet body is shown in Figure 6.5.

The important fields in the message header packet are:

- Version: Contains the version number of the protocol.
- Source Address: Contains the address of the node from which this packet originated.
- Destination Address: Contains the address of the node to which this packet is addressed.

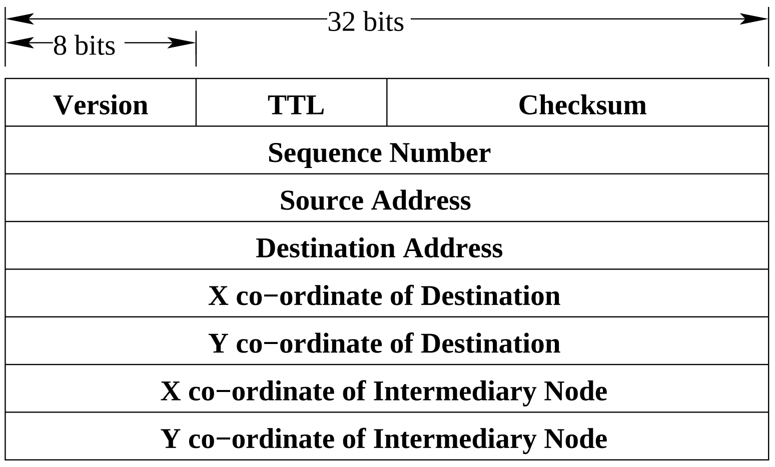

32 bits
8 bits
Version
TTL
Checksum
Sequence Number
Source Address
Destination Address
X co−ordinate of Destination
Y co−ordinate of Destination
X co−ordinate of Intermediary Node
Y co−ordinate of Intermediary Node

Figure 6.4: Format of message header

- X co-ordinate of Destination: Contains the x co-ordinate of the destination node.
- Y co-ordinate of Destination: Contains the y co-ordinate of the destination node.
- X co-ordinate of Intermediary Node: Contains the x co-ordinate of the most recent node along the path.
- Y co-ordinate of Intermediary Node: Contains the y co-ordinate of the most recent node along the path.

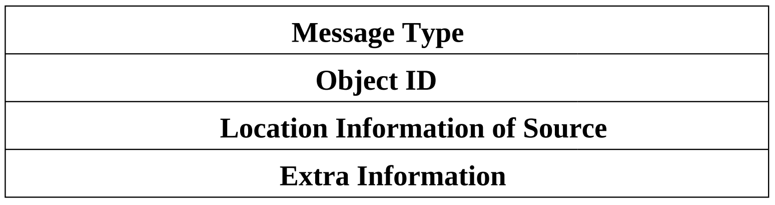


Figure 6.5: Format of message body

The important fields in the message body packet are:

- Message Type: Contains the type of the message.
- Object ID: Contains the ID of the object that is to be registered or discovered.

- Location Information: Contains the location information of the source node from which this packet originated.
- Extra Information: Contains required information that differs according to the message type. In the case when the message type is *REGISTER*, this information contains the location vector of the node. In the case when the message type is *DISCOVER_ACK*, this information contains the list of nodes that own this object along with their location information.

In the event when the Daemon process receives a registry or discovery request, the process checks to see whether it can accommodate this request. The Daemon maintains a table that contains a list of objects registered with this node (along with information about the nodes that own them). In the event of an object registry, information is added to the table, while in the case of an object discovery, the table is queried and the appropriate information is returned. The table consists of a collection of objects each of which is of type *objectEntry*. The data structure of the table and *objectEntry* is given below:

```
typedef struct objectEntry
{
     int objId;                 // Object ID
     NODE nodeList[MAX_VALUE]; // Collection of nodes that own this object
     int nodeIndex;            // Number of nodes that own this object
} OBJ_ENTRY;
OBJ_ENTRY objTable[MAX_VALUE]; // Table of objectEntry's
```

A variable, called *objTableIndex* maintains the current offset in the table and indicates the location where a new element can be added to the table. Any time a new element is added to or deleted from the table, the index *objTableIndex* is appropriately incremented or decremented. To ensure that the table is not full, the index *objTableIndex* is checked every time an element is added to the table. When the table is created for the first time, *objTableIndex* is set to -1.

In the event when a packet is received by the Daemon that is not indented for this node, the Daemon process uses the information in the packet about the location of the destination and the intermediary node to forward the packet. The Daemon uses PILOT to calculate the truncated cone and uses this to create the appropriate IP headers for the packet and sends the packet into the network. The major functions that are used in the Daemon module are provided in Table 6.2.

6.4 SUMMARY

This chapter presented in detail the design and implementation of the proposed service architecture. The design factors involved in designing a working model of the proposed service architecture were elaborated in Section 6.1. The important components of the overall design shown in Figure 6.1 were highlighted and a brief explanation of each component was provided in Section 6.2. Section 6.3 provided a detailed overview of the implementation architecture (shown in Figure 6.3) by detailing the various modules and data structures along with their interactions.

The implementation has tried to closely follow the software engineering principles outlined in Section 6.1.1. The software has been constructed in a way that it is modular. It was developed almost extensively in C using good procedural techniques, except for some sections which were written as part of a test application. Each major component detailed in Figure 6.1 has been implemented as a separate module (as shown in Figure 6.3). Each of these modules export certain pre-specified interfaces which the other modules use to interact with it. An example of this is the shared memory module. It exports the *insert_msg()*, *remove_msg()* and *print_msg()* functions. These functions are used by all the other three modules, namely: registration, discovery and daemon. The code has been written in a manner such that it can be reused over and over again. It was a design concern to try and keep the code as portable as possible. The code has been completely developed in user space, thus enhancing the portability of this implementation. To port this implementation to another platform would only require some fine tuning. Maintenance of the software is very easy because of the high modularity of the software. As long as the interaction between the modules uses only the exported interfaces, any changes in one module will not affect the other modules.

Table 6.2: Important Functions in the Daemon module

Function	**Description**
build_packet()	Build a packet with the appropriate information.
recv_packet()	Receives a packet from the RAW socket.
broadcast_msg()	Broadcasts the given message.
unicast_msg()	Unicasts the given message to the intended destination.
existEntry()	Checks if the given entry is already in the object table.
insertEntry()	Inserts the given entry in the object table.
removeEntry()	Removes the given entry from the object table.
printEntry()	Prints the given entry from the object table.
printTable()	Prints all the entries in the object table.

7.0 EXPERIMENTS AND RESULTS

This chapter details the experimental setup and the analysis of the proposed service architecture and the message forwarding algorithm. Any architecture or protocol that is developed must be tested and evaluated before we can predict the usefulness of the architecture. The remainder of this chapter is organized as follows. Section 7.1 talks about the experimental setup used to analyze the service architecture and the message forwarding protocol. Section 7.2 details the experimental results and their analysis. This section is sub-divided into two sections: Section 7.2.1 details the sensitivity analysis performed on SARA, while Section 7.2.7 details the comparative analysis of PILOT.

7.1 EXPERIMENTAL TESTBED

The service architecture and the message forwarding protocol were implemented in the Glomosim network simulator [36] on Linux and tested under different network conditions. The first set of tests were conducted as part of the sensitivity analysis of SARA. The second set of tests compared the performance of PILOT to GPSR [6], LAR [35] and AODV [46]. We chose to compare PILOT to GPSR, LAR and AODV because, GPSR is a location-based routing protocol that uses greedy forwarding (similar to PILOT); LAR is an on-demand location-based routing protocol and has been used as a benchmark for comparing location-based routing protocols and AODV is an on-demand routing protocol of a different nature.

In order to understand and evaluate the performance of SARA, we performed a sensitivity analysis of the service architecture. We ran extensive simulations for a variety of node densities and node mobilities to study the behavior of SARA under different network settings.

For the second part of the simulation, we did a comparative study by comparing PILOT to GPSR, LAR and AODV in terms of the throughput achieved for a network consisting of mobile nodes by varying different network parameters. The statistics collected for LAR and AODV were available as part of their implementations that are provided in Glomosim. GPSR was implemented in Glomosim for the purpose of our experimental study. For all experiments, the *Random Trip* mobility model was used.

Nodes are assumed to be stationary at the start of the simulation and are associated with the usual limitations on energy and radio communication. We denote the network service area by Λ. In all our experiments, nodes are assumed to be uniformly distributed in Λ. Furthermore, the wireless medium is assumed to be reliable and does not contribute to any packet loss. Each node is assumed to know the uniform hash function used in the network. Also, a node is assumed to know its location and the approximate boundaries of the network service area, Λ. Note that SARA can fully work with logical-coordinates since a node only needs to know its physical or logical locations and the hash function.

To evaluate the *scalability* of SARA and PILOT, we ran simulations of networks of sizes varying between 100 and 1000 nodes. The network simulated was thus varied from a sparsely populated network to a densely populated network. The network service area, Λ spanned a square of size 3000x3000m.

To evaluate the effect of mobility on this architecture, mobility was incorporated into the simulation by using the *Random Trip* mobility model. The most commonly used mobility model for wireless networks is the Random Waypoint model, which is easy to simulate but does not produce realistic scenarios [31]. We used the *Random Trip* mobility model because it is a generic mobility model that achieves realistic scenarios. A tool has been provided by which the Random Trip model can be incorporated for use with the ns-2 network simulator [66]. The tool produces a perfect sample of the node mobility state and can be used as an input to ns-2 [61]. The ns-2 input file was converted to input files that could be used with Glomosim using the tools provided by [3]. During the course of the simulation, some nodes were selected at random to be mobile. Two different node speeds were studied, namely, 5 m/s (a patient being wheeled around in a hospital) and 10 m/s (an emergency situation).

7.2 RESULTS

7.2.1 Sensitivity Analysis of SARA

At the start of every simulation, the *VAFP* is formed and landmarks are fixed. Network nodes are picked at random to form the *virtual registries*. Traffic was generated in the network by simulating object registration and discovery requests. At the start of each simulation, some nodes were chosen at random to register and discover objects.

The metrics that we focused on during this simulation were: impact of the threshold value $\mathcal{K}$ on the number of virtual registries created, latency of object registration and discovery, % of success for object registration and discovery, and the ratio of successful object discoveries versus successful object registrations. Table 7.1 presents a summary of the different design parameters used during the sensitivity analysis portion of the simulation.

Table 7.1: Summary of simulation parameters for sensitivity analysis of SARA

Name	Value
No. of nodes	100 - 1000
Transmission range	250m
Network Size	3000x3000m
Node Mobility	Yes
Mobility Pattern	RANDOM-TRIP
Node Speed	5 m/s and 10 m/s
Node Pause Time	10s
Simulation Time	1H
No of experimental runs	10

Simulation results are presented in the following subsections. In each of the graphs presented in this section, a point represents the average of 10 experimental runs. It is worth mentioning that we were aware of the standard deviation in all simulation runs and we did not encounter a relatively large variance in any of simulations. In the next few sections we will discuss the experimental results.

7.2.2 Number of Virtual Registries

In the first set of simulations, we evaluated the effect of the threshold $\mathcal{K}$ on the number of virtual registries that are created in the network. Having an appreciable number of virtual registries in the network is needed to aid object registration and discovery and mobile node location. This experiment will helps us determine the value of $\mathcal{K}$ that would work well for both static and mobile networks. For the mobile network, the speed of the nodes was set to 10 m/s. The results of the experiments for static and mobile networks are depicted in Figure 7.1 and Figure 7.2 respectively.

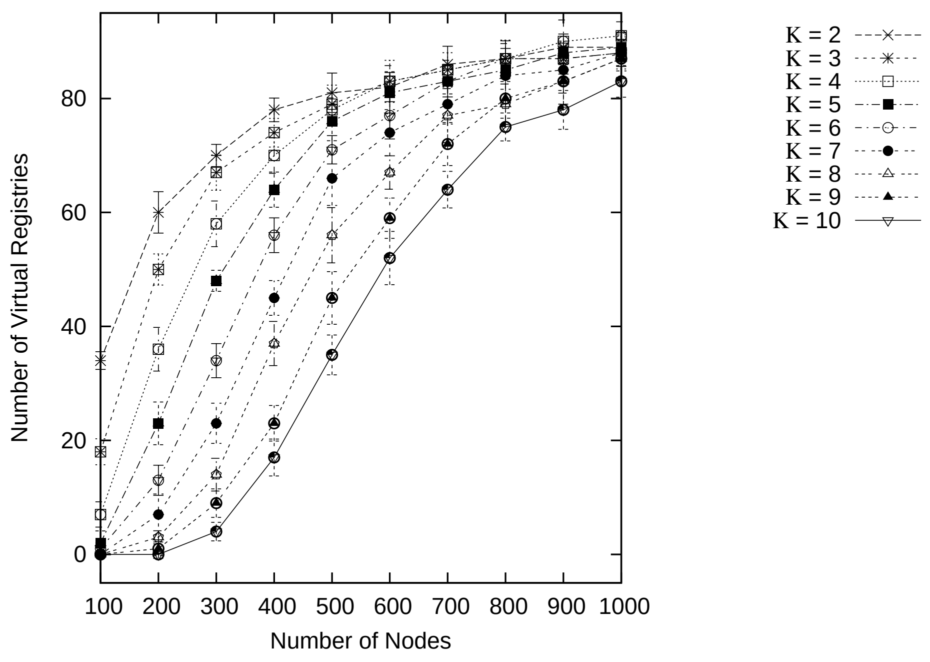


Figure 7.1: Number of Virtual Registries (Static Network)

Figure 7.1 shows that the number of virtual registries in the network does not grow exponentially even when the density of nodes in the network grows. This shows that nodes in SARA do not create new registries if they are already part of an existing one. The number of registries does increase as expected in all cases and is nearly equal when the density of nodes in the network is very high. From the graph, we can observe that the best performance is given when $\mathcal{K} \leq 5$.

From Figure 7.2, we can observe that the number of virtual registries drops when com-

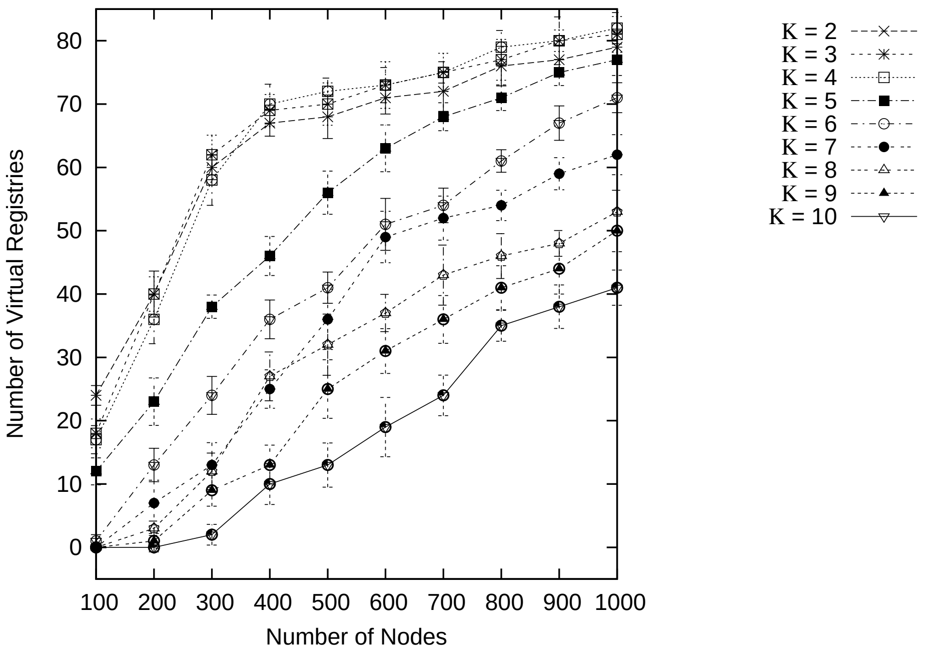


Figure 7.2: Number of Virtual Registries (Mobile Network)

pared to the static case. The presence of mobile nodes in the network leads to some virtual registries disbanding due to the departure of such nodes. Even in this scenario, the number of virtual registries is not very high as the density of nodes increases. This shows that mobile nodes find and join existing virtual registries rather than create new ones. One interesting fact to take from this graph is that, as the network density increases, the number of virtual registries is higher for $\mathcal{K} = 3$ and 4 as compared to $\mathcal{K} = 2$. The reason for this is that having a threshold of 2 results in nodes forming registries with only 2 nodes and such a registry quickly disbands when any one node leaves the registry. Having a very high threshold (e.g: $\mathcal{K} = 10$) results in very few registries, but finding so many nodes in the network to form a registry is not practical. Using these experiments as a benchmark, we set $\mathcal{K}$ to 4 for the rest of the experiments.

The intuition that we take away from this experiment is that having a high value of $\mathcal{K}$ results in fewer registries, but this also leads to a high number of registries disbanding due to the difficulty in maintaining the high number of mobile nodes as members of this registry. Having a very low value of $\mathcal{K}$ ($\mathcal{K} = 2$) leads to more registries being formed in the network, but they also disband and need to re-form even if only one node leaves the registry. A compromise between having a high value of $\mathcal{K}$ and a low value of $\mathcal{K}$ is needed. Having an adaptive scheme where the value of $\mathcal{K}$ is high while recruiting nodes to form a registry, but is lower for the purpose of maintaining the registry would work well in the face of mobile nodes in the network. This results in a scheme where the number of registries in the network is low, but the lifetime of each registry is high.

7.2.3 Latency of Object Registration

In this experiment, we observe the impact of mobility on the latency for object registration (since object discovery follows the same procedure, we only measure the latency for object registration) as the number of nodes in the network increases. Nodes are picked at random and they register a total of 30 objects with the network. The time measured is the time taken for a "successful" object registration. Each point in the graph shows the average time taken across all successful object registration attempts. We compare the performance of SARA in three different cases: a static network, a mobile network, when the nodes move at the speed of 5 m/s and a mobile network, when the nodes move at the speed of 10 m/s. The result of this experiment is shown in Figure 7.3.

Figure 7.3 shows that the difference in latency for a network of static nodes and a network of mobile nodes is not very high. An important implication of this result is that SARA does not impose a large degradation as the mobility of the nodes in the network increases. Another interesting observation is the fact that latency is almost a constant as the node density in the network increases. This observation makes SARA an attractive choice for use by applications expecting some appreciable level of QoS.

7.2.4 Success of Object Registration

For any service discovery architecture, an important evaluation metric is the number of successful registration/discovery requests. For this purpose, we evaluate the % of successful object registrations in SARA as the number of nodes in the network increases. Nodes are

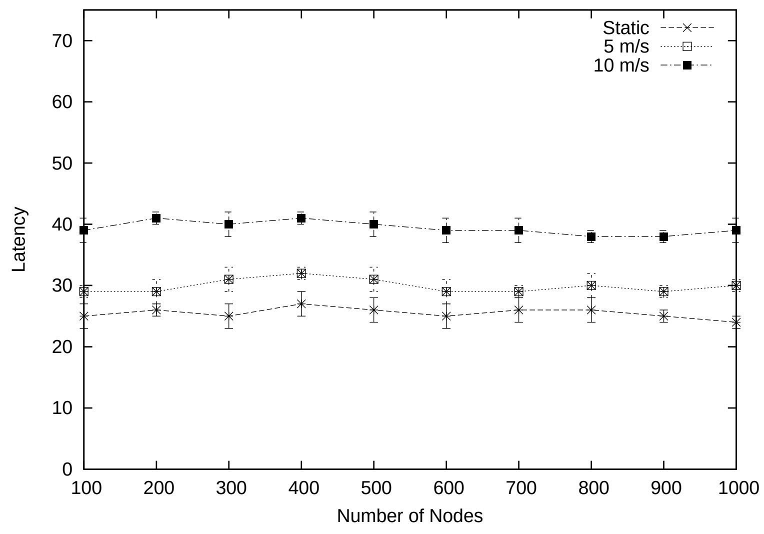


Figure 7.3: Latency of Object Registration

picked at random and they register a total of 30 objects with the network. Each point in the graph represents the percentage out of the 30 objects that were successfully registered with the network. We compare the performance of SARA in three different cases: a static network, a mobile network, when the nodes move at the speed of 5 m/s and a mobile network, when the nodes move at the speed of 10 m/s. The result of this experiment is shown in Figure 7.4.

From Figure 7.4, we observe that in all cases the ratio of successful object registrations increases quickly and finally reaches 100%. This can be attributed to the fact that as the network density grows, the number of virtual registries also increases. This results in the availability of more registries to hold object and node information. The success is always lower for a network of mobile nodes since the mobility of nodes directly impacts the ability to successfully route messages in the network. Also, node mobility impacts the creation and maintenance of the virtual registries in the network. We can also observe from the figure that node mobility does not adversely affect the performance of SARA and the difference is on average less than 20%. As the density of the network increases, the performance of SARA in the mobile case is closer to the performance in the static case.

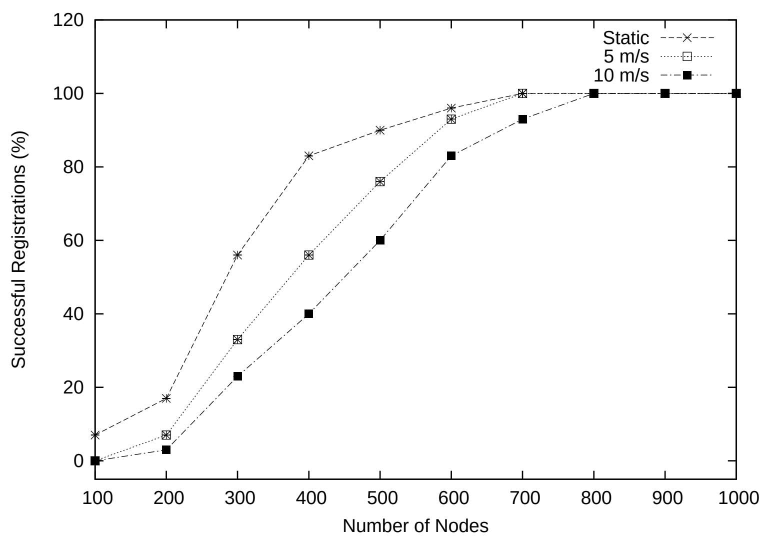


Figure 7.4: Success of Object Registration

7.2.5 Ratio of Object Discovery vs Registration

Another important evaluation metric for a service architecture is the ratio of successful object discoveries with respect to the number of successful object registrations. This metric shows the ability of the service architecture to *discover* the registered objects in the network. For this experiment, while calculating the required ration we only take into consideration the objects that are “successfully” registered with the network. We compare the performance of SARA in three different cases: a static network, a mobile network, when the nodes move at the speed of 5 m/s and a mobile network, when the nodes move at the speed of 10 m/s. The result of this experiment is shown in Figure 7.5.

Figure 7.5 shows that the ratio becomes 1 as the density of nodes in the network increases. This implies that all the objects that are registered with the network are successfully discovered. In the case of low network density, the ratio for the network consisting of mobile nodes is lower than the ratio for the network of static nodes. In a low density network, there is a scarcity of nodes to form and maintain virtual registries to hold node and object information. Also, there is a paucity of nodes that are available to successfully forward messages. As the network density increases, the overall performance also improves and the

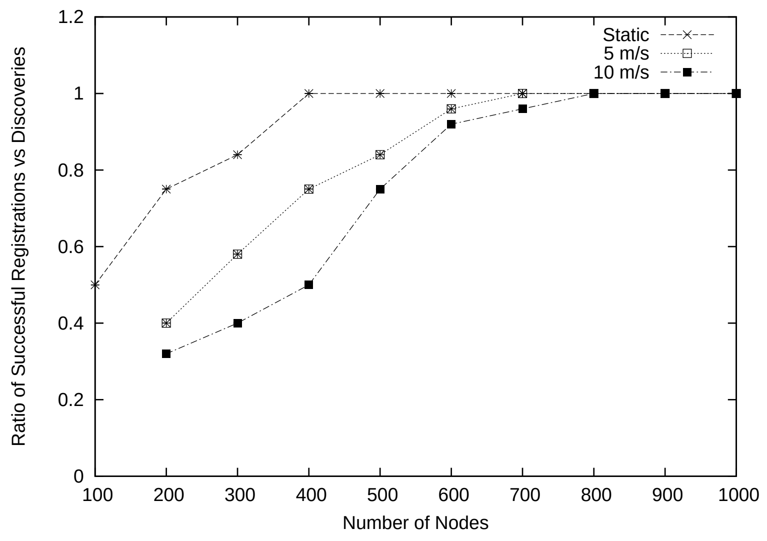


Figure 7.5: Ratio of Object Discovery vs Registration

ratio reaches 1. We can also observe that the ratio for a mobile network is lower (though not by much, on average around 20%), especially in the high mobility case. Node mobility impacts the creation and maintenance of virtual registries in the network. As the density increases, though, the availability of more nodes results in more stable registries and hence the performance improves and finally the ratio reaches 1.

7.2.6 Effect of the Virtual Anchor Forwarding Path

In the next set of experiments, we evaluated the effect of adding the virtual anchor forwarding path to the service architecture. The *VAFP* was modeled as an ordered set of landmark points (physical locations) that spans the network service area. The *VAFP* determines the path to follow to locate the next virtual registry to store/retrieve information object and node information. The *VAFP* is fixed at the beginning of the simulation and does not change through the course of the simulation.

To evaluate the performance of SARA with and without the effect of the *VAFP*, the set of metrics include: % of successful object registrations and ratio of object discovery vs registration. In each of the experiments, the density of nodes in the network was varied from a sparsely populated network (100 nodes) to a densely populated network (1000 nodes). Also, each experiment was performed for a network of static nodes and a network of mobile nodes. In the case of the network of mobile nodes, the speed of the nodes was set to 10 m/s.

In the first set of experiments, we observe the impact of node density on the percentage of successful object registrations for a network with and without the *VAFP*. The results of this experiment for a network with static nodes is provided in Figure 7.6, while Figure 7.7 depicts the result for a network containing mobile nodes.

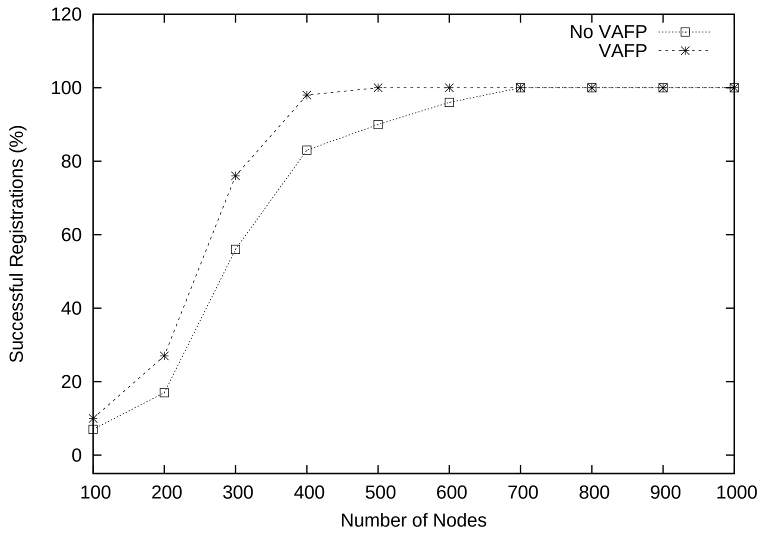


Figure 7.6: Success of Object Registration (Static Network)

From Figure 7.6, we can see that the performance of SARA with the virtual anchor forwarding path is always better and on an average it performs atleast 15 - 20% better than SARA without the *VAFP*. The *VAFP* aids in locating the next virtual registry to register the object information with, which might otherwise have been lost due to the non-availability of a virtual residence at the location of the hash value of the object. The improvement for a sparse network is not that much due to the paucity of neighboring nodes to forward information along the *VAFP* to the next available registry. As the density of the network increases, the performance of the *VAFP* also increases.

Figure 7.7 depicts the result for a network of mobile nodes. It is clear from the graph that the *VAFP* directly results in increasing the success of object registration. Node mobility causes frequent changes to virtual registry membership, thus causing registries to disband.

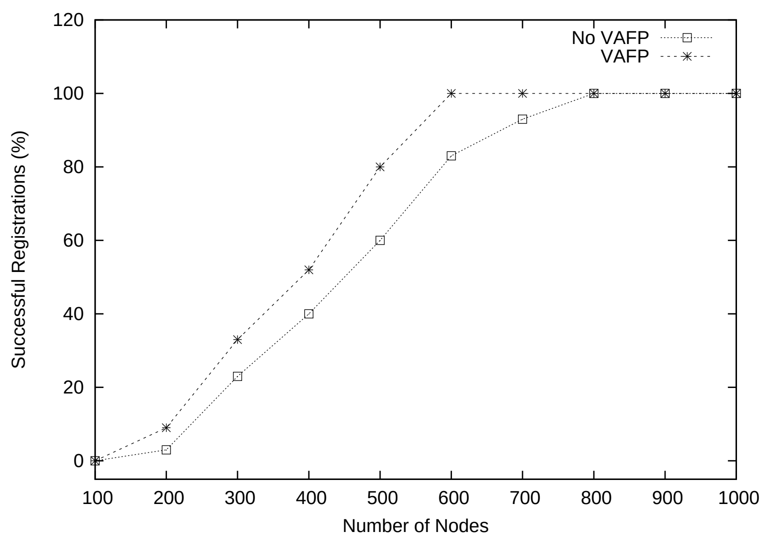


Figure 7.7: Success of Object Registration (Mobile Network)

Using the virtual anchor forwarding path proves effective in locating the next available registry to register the object information with.

In the second set of experiments, we observe the impact of node density on the ratio of successful object discoveries versus successful object registrations for a network with and without the *VAFP*. Figures 7.8 and 7.9 depict the results of this experiment for a static network and a network of mobile nodes respectively.

Figure 7.8 shows that the ratio of object discovery with respect to object registration is higher for the architecture that is augmented with the *VAFP*. This is a direct effect of using the virtual anchor forwarding path to locate and query other available virtual registries in the network for the object information. For a network with a low density of nodes, we see that the performance gain is not very high. This is due to the scarcity of nodes that are available to forward messages along the *VAFP*.

Figure 7.9 shows that the *VAFP* also helps to increase the ratio of object discovery with respect to registration in the case of a network consisting of mobile nodes. The *VAFP* helps in performance gains of around 15-25% on average. As the density increases, the network augmented with the *VAFP* approaches the ratio of 1 faster and ensures that there is no

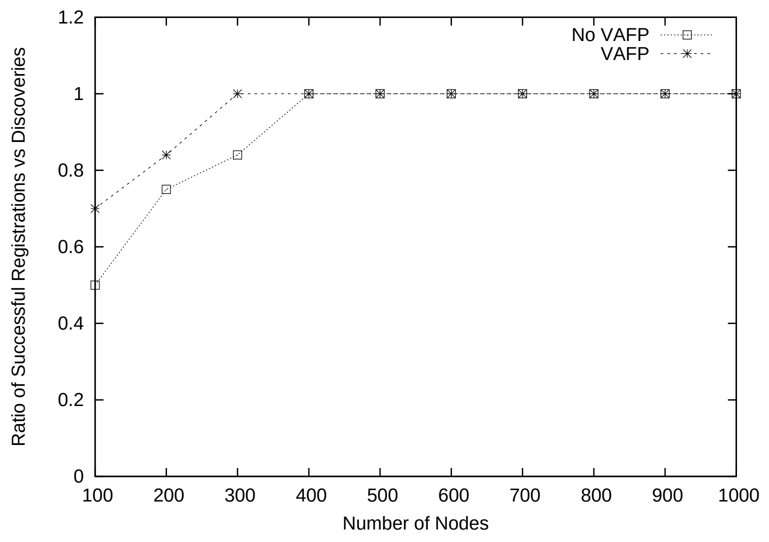


Figure 7.8: Ratio of Object Discovery vs Registration (Static Network)

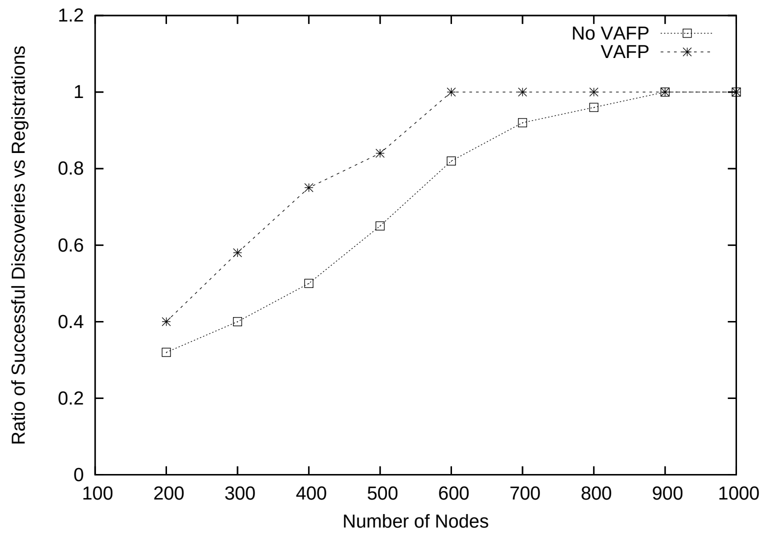


Figure 7.9: Ratio of Object Discovery vs Registration (Mobile Network)

information loss in the network.

The results of our experiments clearly show the benefit of using the virtual anchor forward path to augment the service architecture. In each of the scenarios, SARA with the *VAFP* always outperforms SARA without the *VAFP*. An interesting study that can be performed as part of the future work is to evaluate the structural properties of the virtual anchor forwarding path. Depending on the coverage and geometrical shape of the network service area, the size and structure of the *VAFP* would change. The *VAFP* should be optimized to cover the network service area so as to avoid any "black holes" (areas that can contain virtual registries, but are not covered by the *VAFP*) and also make sure that the *VAFP* is traversed in the correct direction.

7.2.7 Analysis of PILOT

In this section, we perform a comparative analysis of PILOT. Traffic is generated in the network by simulating traffic from a source to a destination. It is assumed that a node has queried for and received the location of the destination and can now send traffic directly to the destination. Traffic generated was CBR traffic with two different sources and two different destinations, to ensure some congestion in the network. Traffic statistics are collected at the destination by measuring the total time taken (in $nano - seconds$) for the packets to reach the destination. At the start of each simulation, some nodes are chosen at random to act as source and destinations. During the course of the experiments, PILOT was compared to AODV, LAR and GPSR.

In all the experiments, we measured the throughput while varying different network characteristics. The metrics that we focused on during this simulation were: impact of transmission range on the throughput, impact of node density on the throughput and the impact of average speed on the throughput. Table 7.2 presents a summary of the different design parameters used in the comparative analysis.

Simulation results are presented in the following subsections. In each of the graphs presented in this section, a point represents the average of 10 experimental runs. It is worth mentioning that we were aware of the standard deviation in all simulation runs and we did

Table 7.2: Summary of simulation parameters for the comparative analysis of PILOT

Name	Value
No. of nodes	100 - 1000
Transmission Range	100 - 500m
Network Size	3000x3000m
Node Mobility	Yes
Mobility Pattern	RANDOM-TRIP
Node Speed	5 m/s and 10 m/s
Node Pause Time	10s
Simulation Time	1H
No of experimental runs	10

not encounter a relatively large variance in any of simulations. In the next few sections we will discuss the experimental results.

7.2.8 Impact of Transmission Range

In the first set of experiments, we evaluated the effect of the node transmission range on the throughput for each of the protocols. The number of nodes in the network was set to 500. The transmission range was varied from 100 - 500m. The experiments were conducted with the average speed of the nodes being 5 m/s and 10 m/s. The result of the experiments are depicted in Figures 7.10 (node speed = 5 m/s) and 7.11 (node speed = 10 m/s) respectively.

From Figure 7.10, we can notice that the throughput is very low for all protocols when the transmission range is very low. This is to be expected, since the number of nodes in the vicinity of the source to forward traffic to the destination is very low, given the low transmission range. We can concur from this experiment that both PILOT and GPSR perform much better when compared to AODV and LAR. Both PILOT and GPSR forward messages using a greedy forwarding mechanism and do not incur the overhead of creating

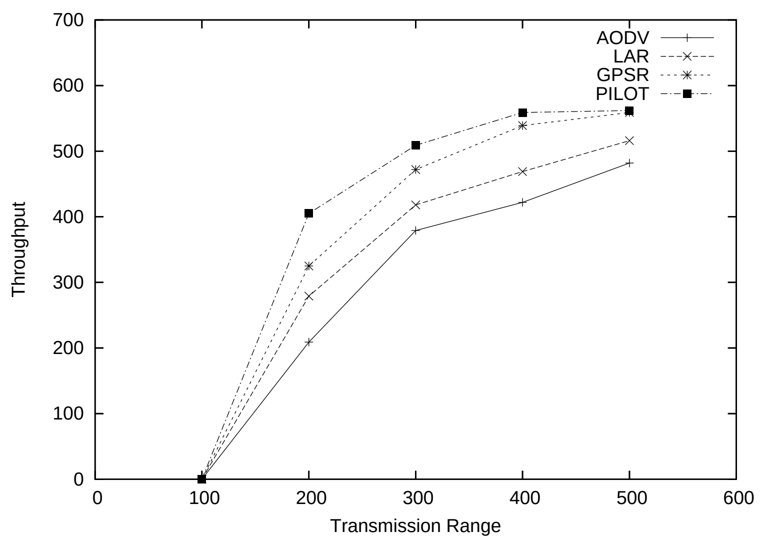


Figure 7.10: Impact of Transmission Range (avg speed: 5 m/s)

and managing routes. As the transmission range increases, PILOT consistently outperforms GPSR too. Unlike GPSR, PILOT does not reach a local maximum and hence it is able to sustain a higher throughput.

From Figure 7.11, it can be seen that PILOT out-performs all other routing protocols. The throughput of all the routing protocols though, is lower than in the previous case when the average speed was 5 m/s. The increased mobility causes frequent changes in the network topology and hence leads to lower throughput. As the transmission range increases, the throughput of all the protocols also increases. This is due to an increase in the number of available nodes in the network that can forward traffic towards the destination. Both PILOT and GPSR out-perform LAR and AODV. This is due to the lower overhead with respect to creating and repairing routes. As before, PILOT performs better than GPSR due to its ability to locate a neighboring node that can forward the traffic towards the destination.

7.2.9 Impact of Node Density

In the next set of experiments, we measure the impact of node density on the routing protocols. This experiment allows us to measure the scalability of the routing protocol. The

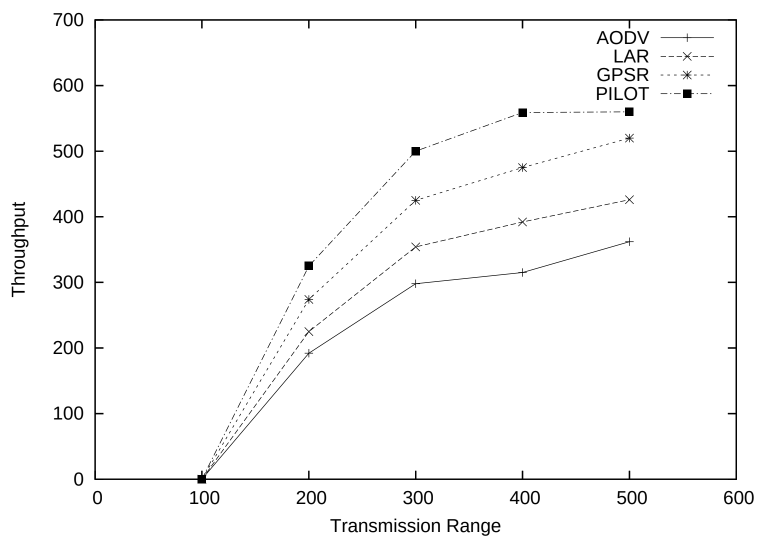


Figure 7.11: Impact of Transmission Range (avg speed: 10 m/s)

number of nodes in the network is varied from 100 to 1000. The network simulated thus was varied from a sparsely populated network to a densely populated network. The transmission range was set to 250m. The experiments were conducted with the average speed of the nodes being 5 m/s and 10 m/s. The result of the experiments are depicted in Figures 7.12 (node speed = 5 m/s) and 7.13 (node speed = 10 m/s) respectively.

From Figure 7.12, we notice that PILOT performs better than GPSR, LAR and AODV. Due to node mobility, routes that were discovered by LAR at the beginning of the simulation may not be valid later and hence another route discovery must be performed. This overhead increases the latency to send packets from the source to the destination. PILOT and GPSR are primarily forwarding protocols and hence they do not incur the cost associated with forming and repairing routes. At 1000 nodes (highly dense network), PILOT achieves the maximum throughput. The denser the network becomes, the better PILOT performs due to the availability of more nodes that can forward the packet towards the destination.

Figure 7.13 shows that the throughput drops slightly when compared to the previous case. This is to be expected due to the higher average speed of the nodes. We notice that PILOT performs better than GPSR, LAR and AODV and at 1000 nodes achieves its

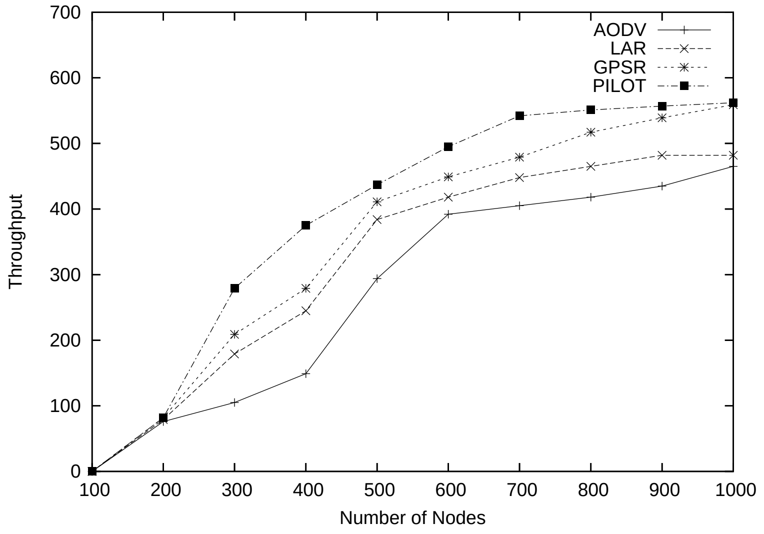


Figure 7.12: Impact of Node Density (avg speed: 5 m/s)

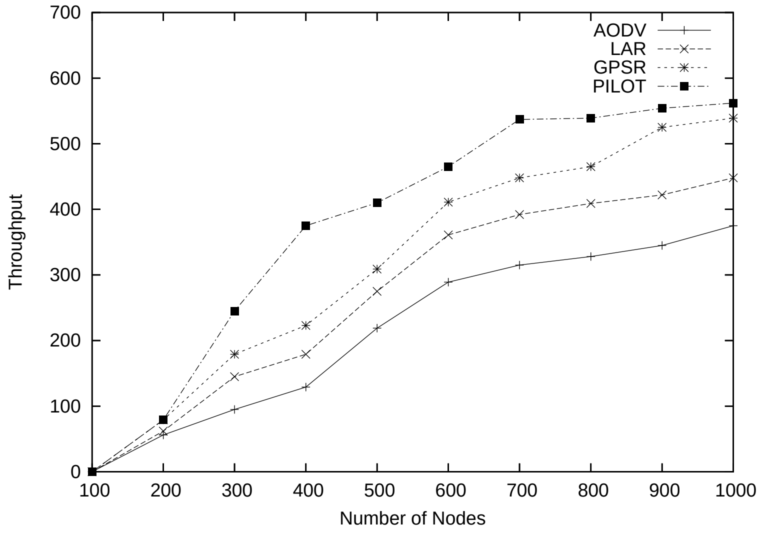


Figure 7.13: Impact of Node Density (avg speed: 10 m/s)

highest throughput. As the number of nodes increases in the network, both PILOT and GPSR perform better due to the availability of more nodes to forward the traffic towards the destination. Also, PILOT outperforms GPSR by more in this experiment as compared to the case when average speed is 5 m/s. experiment. Due to the increased mobility, the neighbor list used by GPSR to decide the next hop node may not be up-to-date. This in turn leads to a degradation in performance of GPSR. PILOT does not incur this overhead since it forwards traffic in a truncated cone-shaped manner and any node within this area (whose probability of forwarding is high enough) can forward the traffic towards the destination.

7.2.10 Impact of Average Speed

In this experiment, we measure the impact of the average speed on the throughput for the different routing protocols. For this experiment the transmission range was set to 250m. The number of nodes is set to 500. The average speed of the nodes was varied from 10 m/s to 50 m/s. The result of this experiment is depicted in Figure 7.14.

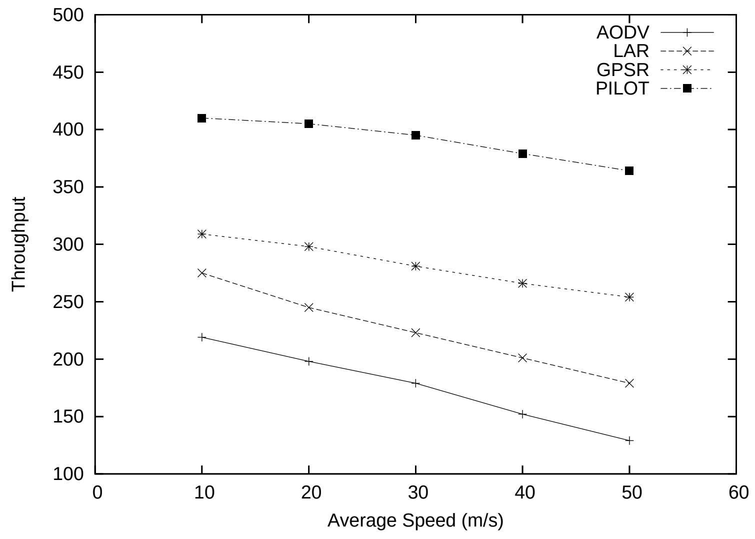


Figure 7.14: Impact of Average Speed

Figure 7.14 shows that, as the average speed of the nodes increases, the throughput goes down in all cases. This can be attributed to the increased mobility in the network due to an increase in the average speed. In this scenario, we can observe that PILOT and

GPSR perform better than both LAR and AODV. This is because, even with increased mobility, there is still a high enough possibility finding a node that can be used to forward the traffic from the source towards the destination. As the average speed increases, the overhead associated with maintaining and discovering routes in the network increases, thus considerably decreasing the throughput for both AODV and LAR. The drop in throughput is not much for both GPSR and PILOT since they do not incur the overhead of creating and maintaining routes.

7.3 SUMMARY

This chapter presented in detail the simulation setup and the analysis of SARA and PILOT. Section 7.1 detailed the experimental testbed used to perform a sensitivity analysis of SARA and a comparative analysis of PILOT. Section 7.2.1 detailed the results of the sensitivity analysis of SARA, while Section 7.2.7 detailed the results of the comparative analysis of PILOT.

The sensitivity analysis of SARA provided us with a method to evaluate the performance of SARA under different network conditions. The virtual registry test showed us that an adaptive scheme (higher number of nodes required to create a registry versus the number required to maintain it) works much better than a scheme that has a fixed value of $\mathcal{K}$. The tests for latency and ratio of object registration versus discovery showed us that SARA is robust and scalable and performs particularly well as the density of the nodes in the network increases. The tests to evaluate the performance of the *VAFP* proved the effectiveness of the *VAFP* since the performance of SARA with the virtual anchor forwarding path was more than 10-20% better than SARA without the *VAFP*.

During the comparative analysis of PILOT, we compared PILOT with GPSR, AODV and LAR. The experiments were conducted while varying the transmission range of the nodes, number of nodes in the network and the average speed. In each of the cases, PILOT and GPSR performed much better when compared to AODV and LAR, especially in the case of high mobility. Both PILOT and GPSR use a greedy forwarding mechanism to forward messages and hence do not incur the overhead of creating and maintaining routes. The results also show that PILOT always outperforms GPSR. Unlike GPSR, PILOT does not need to maintain a neighbor list to select the best possible neighbor node to forward the traffic to. Due to the increased mobility, the neighbor list used by GPSR to decide the next hop node may not be up-to-date. This in turn leads to a degradation in the performance of GPSR. PILOT does not incur this overhead since any node in the truncated cone-shaped area (whose probability of forwarding is high enough) can forward the traffic towards the destination.

8.0 CONCLUSION AND FUTURE WORK

Recent technological advances in engineering and communication have paved the way for a new generation of embedded wireless devices. The capabilities of these devices range from small inexpensive lightweight sensors, with limited memory and computational capabilities, to resource-rich devices, which can support significantly enhanced functionalities. A number of these devices can be deployed on a large scale for sensing and in-situ processing of spatially and temporally dense data, and for carrying out specialized functions. These heterogeneous networks of embedded systems offer the capability to gather critical, real-time information for remote surveillance, collaborative tracking, and distributed control.

The objective of such infrastructureless networks is to support increased mobility, flexibility, and lower cost of managing the resources in comparison to infrastructured networks. These networks are unique, in the sense that a participating device can function as a host as well as a router, thereby dynamically creating paths between network devices. Unlike a fixed wireless network, however, locating a device becomes difficult as users may exhibit high levels of mobility.

Despite advances in areas of routing and media access technology for pervasive and ubiquitous computing, little progress has been made towards large-scale deployment of services and applications in such environments. The lack of a fixed infrastructure, coupled with the time-varying characteristics of the underlying network topology, makes service delivery challenging.

Although typical service discovery frameworks work at a higher layer than routing, it is imperative to integrate service registration and discovery with traffic forwarding. Unlike traditional wireline networks, node mobility is a factor in wireless networks and has to be accounted for while designing the protocols for service registration and discovery.

The basic tenet of this thesis is that service discovery is a key enabling technology to support interactions among heterogeneous devices in ubiquitous and pervasive environments. This thesis takes a unique approach by integrating the different components of a service architecture into a seamless power and location-aware architecture that is well-suited for scalable, robust, large-scale service deployment in pervasive and ubiquitous environments. The contributions of this thesis are:

1. *SARA*: An efficient, robust, scalable and secure framework for large-scale service and application deployment in ubiquitous environments. The basic tenet of this architecture revolves around the concepts of *virtual registries*, *most likely residence* and *location vector*. A virtual registry is a dynamically created administrative domain that enables object registration and discovery. The extent of a virtual registry is such that it encompasses at least $\mathcal{K}$ (threshold) nodes. The information in a virtual registry is maintained by its member nodes. The *most likely residence* of a node is the physical area where the node is likely to be located most of the time. For example, the most likely residence of a fixed node is its physical location. This is registered by the node with the network and is used as a *congregation* point by nodes to contact other nodes. In the case of a mobile node, the node also registers its *location vector*. The location vector of a mobile node is a dynamic time-dependent vector that represents the physical location of the node at a given time, thus reflecting user activity. The *primary* advantage of this approach is that each node can choose to provide its *own* mobility prediction model, which it deems to be most appropriate to its current activity, rather than using a network-wide model which may not be applicable to specific itineraries and situations. Object registration and discovery are achieved by hashing the *object id* to obtain the physical co-ordinates of a *point* (P) within the network service area. The set of mobile nodes in the virtual registry containing P assume the responsibility of maintaining information about the object. The basic *design principle* for our scheme is to use *geographical mapping* for the hashing as opposed to *node mapping* since nodes are mobile. While bootstrapping, a node only needs to know the hash function that is used to register and locate objects in the network. To ensure that there are no *hot spot*s in the network due to hashing, the hash function is chosen to be a *uniform* hash function [12].

2. *PILOT*: A power, resource, and location-aware traffic forwarding algorithm for the service architecture. This algorithm uses a priority based scheme that imposes a priority on the neighboring nodes in a way, such that nodes which are more in line with the direction of the destination have higher probability to forward the message, thereby reducing the delay that traffic suffers on its way towards the destination. The priority is also closely tied to the residual energy-level of the intermediary node to increase network lifetime.
3. Security schemes to ensure user privacy in the proposed service architecture. Three schemes are developed, each of which serve a different purpose and the best scheme can be chosen depending on the requirements and constraints of the network. These schemes are light weight and were developed while taking node mobility and the resource constrained environment into consideration. The proposed security mechanisms are divided into three schemes: *Multiple Location Vector* scheme, *Node-Proxy Based* scheme and *Random-Proxy Based* scheme. All these schemes work on the assumption that each node has a public-key/private-key pair and the identity of a node cannot be ascertained by using its public-key.
4. A proof of concept implementation for the proposed architecture that shows its ability to perform in a real world scenario.

In order to evaluate the performance of SARA and PILOT, we simulated them by implementing and evaluating them using the Glomosim network simulator. We performed a sensitivity analysis of SARA by running extensive simulations for a variety of node densities and node mobilities to study its performance under different network settings. To evaluate the performance of PILOT, we performed a probabilistic analysis and also a comparative analysis by comparing its performance with respect to throughput to GPSR, LAR and AODV for a network consisting of mobile node while varying different network parameters.

The sensitivity analysis of SARA provided us with a mechanism to evaluate the performance of SARA under different network conditions. The tests showed that an adaptive scheme that manages the virtual registry threshold $\mathcal{K}$ is best suited for ubiquitous environments. The performance of SARA is particularly good for networks with higher density due to the availability of more nodes to maintain object and registry information. In the absence of a lot of nodes in the network, the virtual anchor forwarding path provides a useful alternative to ensure that information loss in the network is kept to a minimum.

The probabilistic analysis of PILOT showed that PILOT converges, even when the density of the network is not very high. The probability of a message reaching the destination converges fast for a network where the density of neighboring nodes is high. In the event when the neighboring density is low, the probability can be increased by increasing the angle of the truncated cone (which automatically happens in PILOT, if there are no available nodes in the primary region to forward the message). Using the analysis of PILOT in the presence of network obstacles, it can be seen that the probability of forwarding of PILOT converges as the density of neighboring nodes in the network increases. Even in the case when the obstacle occupies most of the area within the truncated cone, an increase in the angle of the truncated cone results in a higher probability of the message reaching the destination.

The comparative analysis of PILOT with GPSR, LAR and AODV showed that PILOT performed better than all of them, especially so in the case when the density of nodes in the network is high. Both PILOT and GPSR use a greedy forwarding mechanism to forward messages and hence do not incur the overhead of creating and maintaining routes as is the case with LAR and AODV. Unlike GPSR, PILOT does not maintain a neighbor list to decide the best neighbor to forward the packet to. In the event of increased mobility, the neighbor list used by GPSR may not be up-to-date causing a degradation in the performance. PILOT is also power-aware by tying the probability of forwarding to the residual energy of the intermediate nodes.

8.1 FUTURE WORK

Demand for continuous real-time monitoring for critical infrastructure protection and homeland security, disaster management, tether-free health care, and real-time environment monitoring, among many other time-sensitive applications, has become increasingly important. Generally, time-sensitive applications have strict quality of service (QoS) and timing requirements that need to be met to ensure correct functioning of the application. In an emergency situation in the health care sector, for example, delivery of vital signs to the doctor in charge cannot be delayed beyond a specified amount of time in order to avoid irreparable damage to the health of the patient. The ability to select the correct set of devices and services also plays an important part in satisfying the QoS requirements of an application. Given a list of services and their constraints, the best possible service that meets the requirements of the application must be selected. Meeting the performance requirements of these applications depends on the ability of the ubiquitous computing environment to fulfill the demands of the application, even in the presence of failures.

To meet these challenges and achieve an acceptable level of reliability, robustness and fault-tolerance, new paradigms for energy-efficient data management and communication protocols must be studied. In particular, we need to develop (i) models that capture the unpredictable properties of time-critical applications due to failures, (ii) efficient and scalable protocols that allow for data transmission in a timely manner, (iii) robust and efficient protocols to ensure that data fidelity is not compromised, and (iv) selection protocols that select the best service or device available to ensure that the QoS requirements are met.

The solutions investigated must obey the timing and QoS requirements of the application and also take into consideration the resource constrained environment. The approach is to develop novel adaptive schemes which seek to identify the most efficient mode of operation given the context, current network conditions, resource constraints, and timing and QoS requirements of the application.

BIBLIOGRAPHY

[1] A. Gopalan, S. Dwivedi, T. Znati and A. B. McDonald. On the implementation and performance of the (α, t)-Cluster Protocol on Linux. In *Proc. 37th Annual Simulation Symposium*, Apr. 2004.

[2] A. R. Aljadhai and T. Znati. Predictive mobility support for QoS provisioning in mobile wireless environments. *IEEE Journal on Selected Areas in Communications*, 19(10):1915–1930, 2001.

[3] Ad Hoc Network SIMulation. http://www.ansim.info/.

[4] Alex Varshavsky, Bradley Reid, Eyal de Lara. A Cross-Layer Approach to Service Discovery and Selection in MANETs. In *Proc. IEE International Conference on Mobile Adhoc and Sensor Systems Conference (MASS), 2005*, November 2005.

[5] ALOHANet. http://en.wikipedia.org/wiki/ALOHAnet.

[6] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000.

[7] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[8] S. Capkun, L. Buttyan, and J. P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, page 17, 2003.

[9] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46+, 2001.

[10] David M. Goldschlag and Michael G. Reed and Paul F. Syverson. Hiding Routing Information. In *Information Hiding*, pages 137–150, 1996.

[11] Dipanjan Chakraborty, Anupam Joshi, Yelena Yesha, Tim Finin. Toward Distributed Service Discovery in Pervasive Computing Environments. *IEEE Transactions on Mobile Computing*, 5(2):97–112, 2006.

[12] Donald E. Knuth. *Art of Computer Programming, Volume 3*. Addison-Wesley Professional, 1998.

[13] Ed. E. D. Kaplan. *Understanding GPS: principles and applications*. Artech House, Boston, MA, 1996.

[14] E. Guttmann and C. Perkins and J. Veizades and M. Day. Service Location Protocol, 1999.

[15] Endeavour Project, University of California, Berkeley. http://endeavour.cs.berkeley.edu/.

[16] F. Wang, B. Vetter and S. Wu. Secure routing protocols: Theory and practice. Technical report, North Carolina State University, May 1997.

[17] IP Routing for Wireless/Mobile Hosts (mobileip) Charter. http://www.ietf.org/html.charters/mobileip-charter.html.

[18] Francoise Sailhan and Valerie Issarny. Scalable Service Discovery for MANET. *percom*, 00:235–244, 2005.

[19] GNU General Public License (GPL). http://www.linux.org/info/gnu.html.

[20] Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, 7(1):29–58, 2000.

[21] Guanfeng Li, Hui Ling and Taieb Znati. Path Key Establishment using Multiple Secure Paths in Wireless Sensor Networks. In *CoNEXT '05*, 2005.

[22] D. Song H. Chan, A. Perrig. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, may 2003.

[23] H. Pucha, S. M. Das and Y. Charlie Hu. Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks. In *Proc. of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)*, December 2004.

[24] J. C. Haarsten. The Bluetooth Radio System. *IEEE Personal Communications Magazine, pp. 28-36*, February 2000.

[25] Z. J. Haas and M. Pearlman. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. *Internet Draft*, Aug. 1998.

[26] J. P. Hubaux, J. Y. Le Boudec, and M. Vetterli Th. Gross. Towards self-organizing mobile ad-hoc networks: the terminodes project. *IEEE Comm Mag*, 39(1):118 –124, January 2001.

[27] I. Stojmenovic. Home agent based location update and destination search schemes in ad hoc wireless networks. Technical Report TR-99-10, Computer Science, SITE, University of Ottawa, Sep. 1999.

[28] IEEE Standard for Information Technology, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE standard 802.11. Edition, 1999.

[29] Anonymizer Inc. http://www.anonymizer.com/.

[30] J. Li, J. Jannotti, D. De Couto, D. Karger and R. Morris. A Scalable Location Service for Geographic Ad-Hoc Routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, August 2000.

[31] J. Yoon and M. Liu and B. Noble. Random Waypoint Considered Harmful. In *Proc. of IEEE INFOCOM*, 2003.

[32] Jalal Al-Muhtadi, Roy Campbell, Apu Kapadia, M. Dennis Mickunas and Seung Yi. Routing through the Mist: Privacy Preserving Communications in Ubiquitous Computing Environments. In *Proceedings of International Conference of Distributed Computing Systems (ICDCS 2002)*, July 2002.

[33] Jivodar B. Tchakarov and Nitin H. Vaidya. Efficient Content Location in Mobile Ad Hoc Networks. In *Proc. IEEE International Conference on Mobile Data Management (MDM 2004)*, January 2004.

[34] David B Johnson and David A Maltz. Dynamic Source Routing in Ad hoc Wireless Networks. In *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[35] Y. B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad-Hoc Networks. In *Proc. ACM/IEEE MOBICOM*, Oct. 1998.

[36] L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia and M. Gerla. Glomosim: A scalable network simulation environment. Technical Report 990027, UCLA, May, 1999.

[37] L. Cranor, M. Langheinrich, M. Marchiori and J. Reagle. The platform for privacy preferences, 2002.

[38] L. Eschenauer and V. Gligor. A Key Management Scheme for Distributed Sensor Networks. In *ACM CCS2002*, 2002.

[39] Gretchen Lynn. ROMR: Robust Multicast Routing in Mobile Ad-Hoc Networks. *PhD. Thesis, University of Pittsburgh*, December 2003.

[40] M. Weiser. The Computer for the 21st Century. *Sci. Amer.*, Sept. 1991.

[41] A.B. McDonald and Taieb Znati. A Mobility Based Framework for Adaptive Clustering in Wireless Ad-Hoc Networks. *IEEE Journal on Selected Areas in Communications (J-Sac), Special Issue on Ad-Hoc Networks*, 17(8), Aug. 1999.

[42] Dejan S. Milojicic, Vana Kalogeraki, Kiran Nataraja Rajan Lukose, Jim Pruyne, Bruno Richard, and Zhichen Xu Sami Rollins. Peer-to-Peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto, March 2002.

[43] OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/.

[44] Paul F. Tsuchiya. Landamrk Routing: Architecture, algorithms and issues, MTR-87W00174, MITRE, May 1988.

[45] Paul F. Tsuchiya. The Landmark Hierarchy: A New Hierarchy for Routing in very large Networks. In *Proc. ACM SIGCOMM*, August 1988.

[46] C. Perkins and E. Royer. Ad Hoc On-Demand Distance Vector Routing. In *Proceedings 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, 1999.

[47] C. R. Perkins and P. Bhagwat. Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM*, pages 234–244, Oct. 1994.

[48] Pierre-Guillaume Raverdy and Oriana Riva and Agnes de La Chapelle and Rafik Chibout and Valerie Issarny. Efficient Context-aware Service Discovery in Multi-Protocol Pervasive Environments. In *Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)*, page 3, Washington, DC, USA, 2006. IEEE Computer Society.

[49] Portolano: An Expedition into Invisible Computing, University of Washington. http://portolano.cs.washington.edu/.

[50] Project Aura, Carnegie Mellon University. http://www.cs.cmu.edu/~aura.

[51] Project Oxygen, Massachusetts Institute of Technology. http://oxygen.csail.mit.edu/.

[52] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, 2001.

[53] Roberto Rinaldi and Marcel Waldvogel. Routing and data location in overlay peer-to-peer networks. Research Report RZ-3433, IBM, July 2002.

[54] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–??, 2001.

[55] Antony I. T. Rowstron and Peter Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Symposium on Operating Systems Principles*, pages 188–201, 2001.

[56] S. Basagni, I. Chlamtac, V. R. Syrotiuk and B. A. Woodward. A Distance Routing Effect Algirithm for Mobility (DREAM). In *Proc. ACM/IEEE Mobicom*, Oct 1998.

[57] S. C. M. Woo and S. Singh. Scalable Routing Protocol for Ad Hoc Networks. *Journal of Wireless Networks*, 7(5), Sep. 2001.

[58] S. Capkun, M. Hamdi and J. P. Hubaux. GPS-Free Positioning in Mobile ad-hoc Networks. In *HICSS*, 2001.

[59] S. Giordano, M. Hamdi. Mobility Management: The Virtual Home Region. Technical report, EPFL-ICA, March 2000.

[60] S. Helal and N. Desai and V. Verma and C. Lee. Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks, 2003.

[61] S. PalChaudhuri, J. Y. Le Boudec and M. Vojnovic. Perfect Simulations of Random Trip Models. In *Proc. 38th Annual Simulation Symposium*, Apr. 2005.

[62] S. Ratnasamy and B. Karp and L. Yin and F. Yu and D. Estrin and R. Govindan and S. Shenker. GHT: A Geographic Hash Table for Data-Centric Storage in SensorNets. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, September 2002.

[63] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. *ACM Sigcomm*, 2001.

[64] Moy John T. OSPF Complete Implementation. *Addison Wesley*, September 2000.

[65] The Jini Network Technology. http://www.sun.com/software/jini/.

[66] The NS-2 Network Simulator. http://www.isi.edu/nsnam/ns/.

[67] The Universal Plug and Play Forum. http://www.upnp.org.

[68] The Wibree Forum. http://www.wibree.com.

[69] The Wibree Forum Press Release - Wibree joins Bluetooth. http://www.wibree.com/press.

[70] The WiMAX Forum. http://www.wimaxforum.org/.

[71] U. Kozat and L. Tassiulas. Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In *Proc. of IEEE INFOCOM*, 2003.

[72] A. Vaina, M. Amorim, Y. Viniotis, S. Fdida, and J. Rezen de. Twins: A dual addressing space representation for self-organizing networks. *IEEE Transactions on Parallel and Distributed Systems*, 2006.

[73] W. Ye and J. Heidemann and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proc. IEEE INFOCOM*, 2002.

[74] WiMAX - Broadband Wireless Access Technology. `http://www.intel.com/netcomms/technologies/wimax/`.

[75] Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu. Simple Service Discovery Protocol/1.0. *Internet Draft*, Oct. 1999.

[76] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.

[77] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.