A CROSS LAYER PROTOCOL FOR SERVICE ACCESS IN MOBILE AD HOC NETWORKS

by

Mesut Ali Ergin

B.S. in Control and Computer Engineering, İstanbul Technical University, 1999

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Engineering

Boğaziçi University

2003

A CROSS LAYER PROTOCOL FOR SERVICE ACCESS IN MOBILE AD HOC NETWORKS

APPROVED BY:

Assoc. Prof. Cem Ersoy (Thesis Supervisor)	.....
Assoc. Prof. Şebnem Baydere	.....
Prof. M. Ufuk Çağlayan	.....

DATE OF APPROVAL: 27.02.2003

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Cem Ersoy for putting his trust in me throughout this study. Dr. Ersoy gave invaluable advices and helped more than one might think of.

I am very appreciative of working with Şebnem Baydere as a research assistant at the Department of Computer Engineering, Yeditepe University. I owe a lot to her for the academic support she supplied. We would have sure been a part of a better research community, if there had been more people like Dr. Baydere, providing countless opportunities to young researchers.

I present my sincere thanks to Yeditepe University for employing me to take part in SeMA project which was partially supported by The Scientific and Technical Research Council of Turkey (TUBITAK) under grant 101E037/EEEAG-AY-41. I am indebted to all members of The Network Laboratory at Yeditepe University.

There are no words to describe the love, support and patience my friends gave me during this thesis study and beyond. Being afraid of forgetting even one of those great friends, I would never dare to recognize their names here. I believe each of them knows how much I should thank them for making my life so enjoyable.

Without any doubt, members of my family made a lot of sacrifices to make sure I had a perfect study. Actually, nothing was as perfect as them.

ABSTRACT

A CROSS LAYER PROTOCOL FOR SERVICE ACCESS IN MOBILE AD HOC NETWORKS

Mobile Ad Hoc Networks (MANET) are composed of moving wireless communication capable computers usually deployed for the purpose of temporal information exchange in cases where coverage of infrastructured networks is not available. Considering the fragile environment of MANET, simple and application aware communication approaches must be preferred in favor of complex and general purpose cascaded stack of protocols. These approaches must serve the applications need for access to services available on other hosts, addressing the announcement, discovery, binding and utilization of those services.

In this thesis, dynamic access to named non-interactive services in ad hoc networks is studied and a simple cross layer protocol is designed for service discovery and routing. The algorithms of the proposed protocol are implemented in a wireless network simulation software, GloMoSim, for the purpose of algorithm verification and performance evaluation. Some representative applications and scenarios designed out of these applications using the simulation software extensions for the new protocol are also implemented. The results from these experiments have shown that a service aware slim protocol stack implementation is possible for non-interactive service access in mobile ad hoc networks. The advantages of having service awareness in the network layer are also emphasized.

Content of the thesis includes the necessary motivation and background for MANET, proposed communication infrastructure, designed mechanisms, simulation implementation details, experiment and results.

ÖZET

GEZGİN TASARSIZ AĞLARDA HİZMETLERE ERİŞİM İÇİN ÇAPRAZ KATMANLI BİR AĞ PROTOKOLÜ

Gezgin tasarsız ağlar, önceden planlanmış bir telsiz ağ altyapısının kapsama alanında olmaya gerek bırakmaksızın, telsiz iletişim yapma yeteneği olan hareketli bilgisayarların birbirleri ile geçici süreli bilgi değişimi yapmalarına olanak verir. Bu ağ yapısının kırılgan ortamı, basit ve uygulamalardan haberdar iletişim yaklaşımlarının, karmaşık ve genel amaçlı protokol yığınlarına tercih edilmesini gerektirmektedir. Bu yaklaşımlar, uygulamaların sundukları hizmetleri duyurmalarına, ağ üzerindeki diğer hizmetleri keşvetmelerine, bağlanmalarına ve kullanmalarına olanak sağlamalıdır.

Bu tezde etkileşimsiz isimli hizmetlere devingen erişim ile ilgilenilmiş, hizmet keşfi ve yönlendirme sağlamayı amaçlayan basit bir çapraz katmanlı protokol tasarlanmıştır. Önerilen protokoldeki algoritmalar, doğrulama ve başarım ölçümü amacıyla GloMoSim telsiz ağ benzetim yazılımına eklenerek gerçeklenmiştir. Gerçeklenen bu benzetim yazılımı kullanılarak, temsili uygulamalar ve bu uygulamalardan türetilmiş senaryolar oluşturulmuştur. Yapılan bu deneylerden edinilen sonuçlar, etkileşimsiz isimli hizmetlere devingen erişim için hizmetlerden haberdar bir basit protokol yığını kullanımının mümkün olduğunu göstermiştir. Ayrıca ağ katmanı seviyesinde, hizmetlerden haberdar olmanın getirileri vurgulanmıştır.

Tezin içeriğinde gerekli hazırlık bilgileri, önerilen iletişim altyapısı, tasarlanan düzenekler, benzetim yazılımı ayrıntıları, deneyler ve sonuçlar bulunmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

ABR	Associativity Based Routing
ACK	Acknowledgement
AODV	Ad Hoc On-demand Distance Vector Routing
API	Application Programming Interface
CBR	Constant Bit Rate
CEDAR	Core-Extraction Distributed Ad Hoc Routing
CGSR	Clusterhead Gateway Switch Routing
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear To Send
DA	Directory Agent
DBF	Distributed Bellman-Ford
DCF	Distributed Coordination Function
DDR	Distributed Dynamic Routing
DHCP	Dynamic Host Configuration Protocol
DIFS	DCF Inter Frame Space
DSDV	Destination Sequenced Distance Vector Routing
DSR	Dynamic Source Routing
DSR	Domain Space Resolver
DSSS	Direct Sequence Spread Spectrum
ETSI	European Telecommunications Standards Institute
FAMA	Floor Acquisition Multiple Access
FIFO	First In First Out
FSR	Fisheye State Routing
FTP	File Transfer Protocol
GloMoSim	Global Mobile Information Systems Simulation
GPRS	General Packet Radio Service
GSR	Global State Routing
HSR	Hierarchical State Routing

HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
INR	Intentional Name Resolver
INS	Intentional Naming System
IP	Internet Protocol
ISM	Industry Science and Medicine
JVM	Java Virtual Machine
LAR	Location Aided Routing
MAC	Medium Access Control
MACA	Medium Access with Collision Avoidance
MANET	Mobile Ad Hoc Network
MPEG	Moving Pictures Expert Group
MPR	Multi Point Relay
OLSR	Optimized Link State Routing
OSI	Open System Interconnection
OSPF	Open Shortest Path First
PCF	Point Coordination Function
PDA	Personal Digital Assistant
PDR	Packet Delivery Ratio
PDU	Protocol Data Unit
QoS	Quality of Service
RFC	Request For Comments
RPGM	Reference Group Mobility Model
RTS	Request To Send
SA	Service Agent
SDP	Service Discovery Protocol
SIFS	Short Inter Frame Space
SLM	Salutation Manager
SLP	Service Location Protocol

SSDP	Simple Service Discovery Protocol
SSDS	Secure Service Discovery Service
SSR	Signal Stability-Based Adaptive Routing
TCP/IP	Transmission Control Protocol/Internet Protocol
TORA	Temporally Ordered Routing Algorithm
TTL	Time To Live
UA	User Agent
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
VAD	Voice Activity Detection
VBR	Variable Bit Rate
VoSeMA	Voice Over SeMA
WAP	Wireless Access Point
WLAN	Wireless Local Area Network
WRP	Wireless Routing Protocol
XML	Extensible Markup Language
XSD	XML Schema Document
ZHLS	Zone-Based Hierarchical Link State Routing
ZRP	Zone Routing Protocol

1. INTRODUCTION

Increasing use of small and mobile computing devices, such as personal digital assistants (PDA) and cellular phones, imposes the need for much simpler approaches to communication methods. This is because wireless communication bandwidth, available power (both for computing and wireless communication) and other interface devices do not resemble their wired and fixed world counterparts, such as desktop computers. With the introduction of mobility into communication, static configuration for any resource on the network becomes costly and sometimes impossible.

Ad hoc networks have started to appear in computer communications world as a result of the temporal need for 'on-the-fly' data exchange among those computing devices equipped with wireless communication hardware. In this sense, ad hoc networks differ from other wireless networks such as cellular networks, satellite networks or even wireless local area networks (WLANs). An ad hoc network is formed without any external aid to construct an infrastructure and may require multiple hops to connect some of the hosts in the network. Apart from the ad hoc nature of the communication, there are three major key issues that characterizes ad hoc networks. They are varying wireless link connectivity, host mobility, and energy constraints.

Since an ad hoc network can be rapidly deployed and reconfigured on an environment without a previously studied infrastructure, military applications were the most promising potential field of use until recently. With the increasing need of (anytime, anywhere) access to information, commercial applications of ad hoc networks have started to appear. A very recent increase in interest to ad hoc communication is related to the field of *sensor networks*, where small embedded devices autonomously try to form a network to disseminate sensed information towards a destination [1].

There are numerous research activities in wireless networks, and ad hoc networks form one major research track among them. Although number of diverse wireless and mobile technologies increase, they tend to serve the sole purpose of mobile, unified and

continuous access to networked services, which is the essential motivation behind the *ubiquitous* [2] or *pervasive computing* [3].

1.1. Problem Definition

Ad hoc networking has its name because of the style of communication need between users of the network. Therefore it does not refer to a specific technique or technology at a given layer, such as gigabit ethernet networking or fiber optic communication, but is a multi-layer problem. Although ad hoc networking related research frequently focuses on fundamental problems like routing, channel access etc., the problems of concern are spread all over the classical network protocol stack. Communicating parties usually do not have access to central, structured services but they try to dynamically adapt themselves to the current environment, usually via assuming some others presence and will for participation. This mood of communication creates hard problems to solve compared to the classical wired networking world.

An important trade off is of concern where proposed solutions try to improve an isolated part of the classical networking stack for ad hoc networking purposes. Using such an approach, there exists the risk of inheriting structured network assumptions from those classical solutions. But this kind of an approach is usually better when compatibility with existing protocols is needed. For example, a routing protocol trying to route IP datagrams in an ad hoc network assumes that IP is valid and applicable as a network layer (or as further transport services) in ad hoc networks. Although IP may be optimized for the purpose, any other possible alternatives are to be discarded to keep routing compatible.

Since multiple access wireless link protocols are more mature (in the sense that there are many commercially available ad hoc communication capable wireless local area network interface cards) than ad hoc network protocols at layers of network and above, more problems are identified and attacked at these upper layers. A course classification of ad hoc networking problems may be done in two categories as routing and application.

1.1.1. Routing Related Problems

In [4], Tanenbaum defines routing as the decision on an incoming packet specifying the output line (or the next hop) the packet should be transmitted. Even definition itself has the implicit assumption of the knowledge of the topology, at least a part of it. In networks without certain infrastructure information available, this problem is usually solved by using topology discovering messages (e.g., HELLO packets). In ad hoc networks, however, topology information maintenance for routing is much more costly because nodes are mobile, and wireless links are fragile compared to wired links, even with static nodes. Changing topology requires an ad hoc network node to have up-to-date information, at least for some part of the current topology by either a proactive or reactive approach as mentioned in Section 3.1. The routing protocol for an ad hoc network should optimize the amount of the network bandwidth and node energy used for topology discovery and maintenance purposes and should be fast enough to gather necessary topological information to determine how to carry on routing of a packet towards a specific destination node.

One major problem to be addressed by the routing algorithm is the scalability of the mechanism. The routing algorithm performance should stay in reasonable boundaries as the number of nodes and size of geographic area of concern increase. Having well-scaling algorithms for large number of mobile nodes is not trivial. Factors that have impact on the scalability are not only internal features of routing algorithms but also link layer properties of nodes (i.e., CSMA type medium access has its own issues of scalability). A routing algorithm will scale better if it is built on distributed control algorithms, rather than global information from all over the network. The routing algorithms ability to scale to hundreds of nodes may determine its wide use in the future.

Another problem that the routing algorithm will be faced with is its complexity. As the decision of a relaying node in the ad hoc network is more dependent on externally gathered information, the complexity of the routing protocol increases. It will be easy to implement a simpler routing protocol and its computational requirements (both

processor and memory wise) will be low comparably. The routing algorithm should also be kept as simple as possible so far as specialized nodes are concerned. Specialized nodes perform rather critical operations of routing (as cluster heads, group leaders or zone boundary nodes) and yet they are prone to link and other failures like other nodes of the network. In the extreme case, routing is aided or performed by some central node, equipped with up-to-date topological information. This approach is usually not preferred except for some small ad hoc networks. Therefore routing protocol should keep homogeneity of the ad hoc network (in the sense that duties of nodes are identical) as much as possible. If heterogeneity is unavoidable, failure resilient distributed duty assignment algorithms must be provided to have routing service available at all times.

An important feature that may affect routing decision of a specific node is the knowledge of available capabilities of the nodes on the way to the destination node. For example, using the smallest hop including path for routing all packets to a destination may cause a bandwidth bottleneck on the network depending on the current topology (i.e., increased medium access contention for the hosts on route). Additionally, if power efficiency is considered, this approach will also cause unfair power consumption for the nodes on route, resulting in quickly dying critical nodes in the ad hoc network. More examples may be given in favor of *feature-aware routing* mechanisms. Therefore a routing mechanism for an ad hoc network should provide ways to use node features as decision criteria in routing.

Fundamental problems that should be addressed by ad hoc network routing algorithms are not limited to those mentioned above. Many others, like multicast routing and secure routing, are vital for different applications of ad hoc networks and have still open issues to be investigated for researchers.

The MANET working group of IETF [5] is primarily focused on developing and evolving routing specifications for ad hoc networks. As the time of writing, group has already published ten draft routing protocols and one RFC on evaluation of routing protocols.

1.1.2. Application Related Problems

Requirements of applications in ad hoc networks have significant impact on the solutions that should be provided by any ad hoc networking protocol. Applications to be developed and run on ad hoc network hosts can not be designed like their *wired-world* counterparts. Powerful processing, plenty of storage space, and enhanced device features (e.g., large high resolution display units) may and would probably be unavailable. Also, *QoS* (i.e., upper limit on packet loss, guaranteed end-to-end rates or delay) is not an easy to offer feature for ad hoc networks, considering limited wireless spectrum (providing limited available bandwidth), mobile users and dynamic topology, time and environment varying weak wireless links. Therefore an ad hoc network application should be aware of the underlying environment to some extend and adjust itself accordingly considering the services and capabilities that are available. This will lead to better performing adaptive applications having ad hoc aware design from the beginning.

Applications should be provided natural mechanisms to suit themselves to the dynamic characteristic of the ad hoc network. A major mechanism serving this purpose is definitely (late) binding to services in the ad hoc network. Having well defined algorithms for discovering, binding and utilizing services on the ad hoc network is important, since the main reason of existence for an application is to benefit from a service provided by another host over the ad hoc network (which may even be the routing itself). For the same reason, a descriptive mechanism to represent these services is also needed. Applications will make use of this representation of services to decide if a candidate service is the one satisfying requirements.

Additionally, applications should have their end-to-end communication services from lowest possible layers, since every new introduced layer into the ad hoc networking stack will bring its overhead and complexity to this already fragile environment. Therefore better performing applications will require slim network stacks and benefit from cross-layer design of these kind of stacks. This idea of cross-layer design is further described and discussed in [6].

1.2. Motivation

The fundamental motivation behind the work in this thesis is to provide applications of ad hoc networks with a simple cross-layer protocol stack that is capable of offering mechanisms to announce, discover, bind to and use the services made available by all other applications on the network. This proposed service-centric approach tries to define all needed algorithms to have communicating applications. Required underlying connectivity may be supplied by many of the available wireless link layer protocols.

Using some of the currently available ad hoc network protocols together may provide this defined functionality, such as using TCP/IP protocol stack with a suitable IP datagram routing approach (e.g., DSR or AODV) and a service discovery protocol (e.g., SLP). Such available protocols are explained in detail in Chapter 3. However, using these well-studied 'heavy-weight' protocols may not be suitable for many applications so far as the complexity of those protocols are concerned. Such protocols are usually variants of their wired and fixed world counterparts, and bring their 'abstraction from neighboring layers'. In order to have an adaptive communication capability in a dynamic environment (i.e., ad hoc networking environment), the available protocols should be kept as simple as possible with cross-layer design in mind. A cross-layer design, in this sense, aims to provide direct service to applications without any middleware layers and their protocols (i.e., such as without having separate network and transport layers to create lower layer independency). This characteristic is needed to let the applications suit themselves to the underlying dynamic network without using 'all-purpose' network protocols.

The proposed simple approach in communication of ad hoc network applications is designed with *service access* as the objective. Having service access as the driving force of the proposed approach is natural because the applications are built to benefit from the services available on the ad hoc network. For this reason, service awareness is integrated into the algorithms of the protocol.

1.3. Contribution

In this thesis, a simple cross-layer protocol is designed for use of ad hoc network applications. This protocol has necessary algorithms to announce, discover and bind to network services. While providing service access for hosts, the protocol also offers solutions to host addressing, multihop packet routing, session and buffer management, and service naming.

Using the proposed protocol, applications may use *non-interactive services* (See Section 4.1 for details) available on the ad hoc network. The protocol algorithms are designed as simple as possible to help building an easy to implement network stack and only assume a basic link connectivity from underlying data link layer protocol. Applications using this protocol do not need any other protocols to have transport service or service discovery service.

For the algorithms of the protocol to work in a service-aware manner, a service definition model is designed using extensible markup language (XML). This *attribute-value* pair holding design is used throughout the algorithms of the protocol to refer specific instances of services offered by the hosts of the ad hoc network. Hosts of the ad hoc network are also represented as attribute-value pair holding XML instances.

Having host and service instances used in algorithms, an easy to enhance protocol is designed. Features like *battery awareness* or *QoS provisioning* may be implemented to be embedded into the proposed protocol.

1.4. Organization of the Thesis

This chapter of the thesis gives an insight into the proposed protocol. Rest of the chapters are organized as follows. In Chapter 2, basic definitions, medium access issues and mobility models are presented to form a background on the subject. Chapter 3 briefly visits the state of the art in ad hoc networks, emphasizing routing and service discovery related work. Chapter 4 gives the details of the proposed protocol by

explaining protocol elements and algorithms developed. Chapter 5 explains the inner workings of the simulation implemented and Chapter 6 presents the experiments done and their results. Concluding remarks and future work are given in Chapter 7.

2. BACKGROUND INFORMATION

2.1. Definitions

In this section, we give the definitions of frequently used concepts throughout the thesis. If possible and available, we provide sources of definitions and alternative approaches for definitions.

2.1.1. Ad Hoc Network

Ad hoc is a Latin origin adjective meaning 'arranged', 'for the particular purpose' [7]. An *ad hoc network* or to be more clarifying, a *mobile ad hoc network* is "an autonomous system of mobile routers (and associated hosts) connected by wireless links –the union of which form an arbitrary graph" as defined by Internet Engineering Task Force (IETF) Mobile Ad Hoc Networks Official Charter [5]. They use abbreviation *MANET* for mobile ad hoc network and further describe it as follows: "The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet."

A MANET is also defined as "a network architecture that can be rapidly deployed without relaying on pre-existing fixed network infrastructure" in [8]. With the two definitions in mind, characteristics of an ad hoc network may be summarized in the following (non-exhaustive) list:

- wireless connectivity (usually bandwidth constrained)
- mobility of hosts
- lack of inherent infrastructure
- multi-hop connections (participation for *others*)
- rapid deployment, short term usage
- self configuration, reconfiguration

- adaptiveness under node and link failures
- energy constraint operations
- limited physical security
- expected to scale well in most cases

Ad hoc networks find use in diverse applications for tactical operations, rescue missions, law enforcement and education. Although establishing a base requirement analysis for all those types of applications is hard, it is obvious that applications of ad hoc network require an immediately deployed data network for cooperative information exchange. In [6], Goldsmith and Wicker discuss how flexible must an ad hoc network be for such diverse applications and they categorize applications as data networks, home networks, device networks, sensor networks and distributed control systems.

Ad hoc networking is an interesting research area where one may easily find the restrictions that are available on various kind of networks appearing altogether. In any given ad hoc network, nearly all resources are scarce such as links are low capacity, nodes are battery powered, connections are dependent on others presence and will etc. This makes ad hoc networking research both interesting and demanding.

2.1.2. Other Definitions

Following list consists of some of the terms used and their short descriptions to provide clear understanding for the rest of the text. Some of the definitions are taken from [9].

- **Access Point**: A two port bridge that connects a wireless LAN to a wired Ethernet LAN. Also known as Wireless Access Point (WAP).
- **Announcement**: Process of emitting an information to the network via broadcasting or flooding.
- **Asymmetric Link**: A link with transmission characteristics which are different depending upon the relative position or design characteristics of the transmitter and the receiver of data on the link. For instance, the range of one transmitter

may be much higher than the range of another transmitter on the same medium.

- **Broadcast**: The delivery of data to every node on a link (i.e., within range of the transmitter).
- **Channel**: The center frequency that the wireless device uses to transmit.
- **Control Message**: Information passed between two or more network nodes for maintaining protocol state which is not associated to any specific application.
- **Flooding**: The process of delivering data or control messages to every node within the ad hoc network.
- **Forwarding Node**: A node within an ad hoc network which performs the function of forwarding packets from one of its neighbors to another.
- **Hidden Terminal Problem**: The problem whereby a transmitting node can fail in its attempt to transmit data because of destructive interference which is only detectable at the receiving node, not the transmitting node.
- **Link**: A communication facility or physical medium that can sustain data communications between multiple network nodes, such as an Ethernet (simple or bridged).
- **MAC Layer Address**: An identifying address (sometimes called the link layer address) associated with the link interface of a node on a physical link.
- **Neighbor**: A neighbor is any other node to which data may be propagated directly over the communications medium without relying the assistance of any other forwarding node.
- **Next Hop**: A neighbor which has been designated to forward packets along the way to a particular destination.
- **Pathloss**: A reduction in wireless signal strength caused by traversing the physical medium constituting the link.
- **Payload**: The actual data within a packet, not including protocol headers which were not inserted by an application.
- **Route Discovery**: The process of finding and setting up a route between a source and a destination
- **Scalability**: Wide applicability of a protocol to large as well as small populations of nodes participating in the protocol.

- **Scenario**: A described course of actions in a sample ad hoc network, specifying host population, environment, mobility, applications etc.
- **Service Discovery**: The process of finding a network service provided by some other host of the network.
- **Source Route**: A source route from node A to node B is an ordered list of host addresses, starting with the host address of node A and ending with the host address of the node B. Between A and B, the source route includes an ordered list of all the intermediate hops between A and B, giving each nodes host addresses.

2.2. Medium Access Layer

Medium Access Protocols for wireless networks deal with the sharing of available spectrum among users. To address this, the spectrum is first to be divided into channels. This division may be in the form of frequency division, time division, code division or a combination of those three. Then there is the issue of assignment of those channels to user. Since most data users will not be requiring continuous transmission, dedicated channel assignments are avoided. For this purpose, random access methods are widely used in accessing available wireless channel. Further discussion on the topic is found in [10]. In the following subsections, two characteristic problems for medium access are given and some of available MAC protocols are listed. Among those, most widely used IEEE 802.11 is inspected in some more detail.

2.2.1. Fundamental Medium Access Problem

Among the available MAC protocols suitable to be used in wireless ad hoc networks, *Carrier Sense Multiple Access* (CSMA) [11] is widely used as the channel access strategy. Several MAC protocols based on CSMA are proposed, eliminating the two basic problems of the CSMA type access, *hidden* and *exposed* terminal problems.

Hidden and exposed terminal problems are illustrated in Figure 2.1. The hidden terminal problem occurs because there exists no transition property between wireless

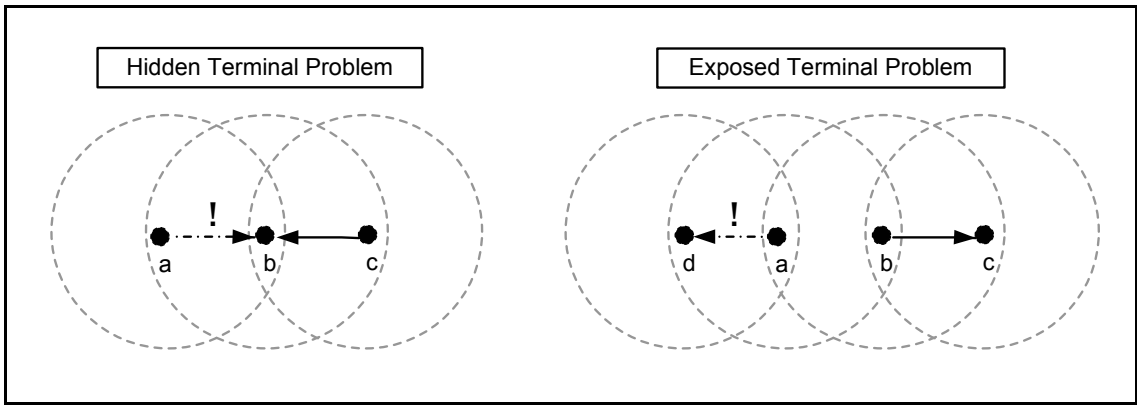


Figure 2.1. Hidden and exposed terminal problems

communication ranges of nodes. *Node a* hearing *node b* and *node b* hearing *node c* does not necessarily mean that *node a* hears *node c*. In the hidden terminal problem part of Figure 2.1, *node c* is communicating with *node b*. Since *node a* is not in the transmission range of *node c*, it can not hear the session between *node c* and *node b*. Therefore whenever *node a* wants to communicate with *node b*, it will sense the medium free and ruin the session between *node b* and *c*, by causing collisions at *node b*. For the exposed terminal problem example, *node b* is transmitting data to *node c*. Since *node a* hears this transmission, it defers from accessing the medium. However, *node a* wants to transmit data to *node d*, which does not cause collisions at *node c*. Although it is feasible for both transmissions to take place simultaneously, CSMA access scheme does not permit this transmission. The reason for both of these problems is the fact that collisions occur at the receiver, while the CSMA protocol checks the status of the medium at the transmitter [12].

In a shared wireless communication medium, nodes can not detect a collision while transmitting, since node's own transmission power is always dominant compared to the power of signal received. Therefore *collision avoidance* techniques are used in wireless environments. Solutions to the mentioned hidden and exposed terminal problems usually include a dialog between the transmitting and the receiving node that announces the upcoming transmission. This mechanism is called *Request To Send* and *Clear To Send* (RTS/CTS) mechanism and provides avoidance from collisions. It is used in MAC protocols like MACA [13], MACAW [14], FAMA [15], EYNPMA [16],

and IEEE 802.11 (CSMA/CA) [17].

Among the protocols that support wireless channel access, IEEE 802.11, ETSI HIPERLAN/2 [18], HomeRF [19], and Bluetooth [20] are standardized protocols. Following subsection provides a closer look to the widely used IEEE 802.11 MAC protocol.

2.2.2. IEEE 802.11 MAC Protocol

IEEE 802.11 is a standard protocol for wireless local area networks (WLAN) with physical layer (PHY) and medium access control layer (MAC) specifications [17]. It provides asynchronous and time bounded delivery service for wireless connectivity of fixed, portable and mobile stations moving at pedestrian and vehicular speeds within local area. It also defines an ad hoc network for devices within mutual communication range. IEEE 802.11 provides maximum data rates of 1.2 Mb/s (802.11), 11 Mb/s (802.11b), 54 Mb/s (802.11g and 802.11a). All current IEEE 802.11 variants work at 2.4 GHz ISM band except 802.11a which operates at 5 GHz band (only licensed for North America).

IEEE 802.11 is also an interoperability standard for wireless LAN devices that identifies three major distribution systems for wireless data communication:

- Direct Sequence Spread Spectrum (DSSS) Radio Technology
- Frequency Hopping Spread Spectrum (FSSS) Radio Technology
- Infrared Technology

The 802.11 MAC specification provides shared access to a wireless channel by making use of two access methods called *point coordination function* (PCF) and *distributed coordination function* (DCF). PCF access method is not of our interest so far as ad hoc networking is concerned, because it is used by the access points of WLANs (Wireless Access Point –WAP) to coordinate contention-free access of neighboring nodes to the medium. Access points are devices used to construct an infrastructured network (possibly connected to a wired backbone) where each and every host of the

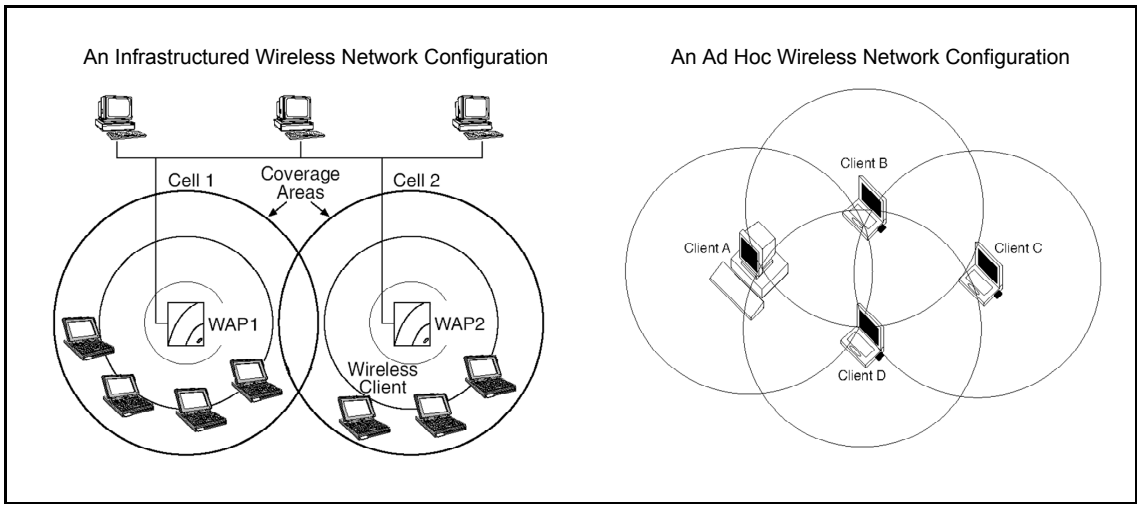


Figure 2.2. An infrastructured and ad hoc wireless network configuration

network should be in the wireless coverage area of the WAP as illustrated in Figure 2.2.

DCF, however, is used in wireless networks without an access point. Hosts communicate with each other as the definition of ad hoc networking states. This mode of communication is also referred to as *ad hoc* mode in 802.11 specification. An ad hoc network configuration example is also given in Figure 2.2. DCF access method is based on CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) and provides contention-based shared access to the medium.

The main access mechanism method used in DCF is called *Basic Access Method*. Before a station starts transmitting a MAC protocol data unit (PDU), it has to sense the channel to be idle. Channel is assumed to be idle if no activity is sensed for a time interval of DIFS (DCF Inter-Frame Space). After the transmission of the station takes place, the receiver should acknowledge the frame by an ACK frame, since the sender may not hear the collisions at the receiver which prevents faithful delivery as explained in Section 2.2.1. To transmit an ACK, the receiver waits for another time interval of SIFS (Short Inter-Frame Space) to ensure the channel is idle. If the transmitter does not receive this expected ACK frame in a certain time-out period, it presumes the data frame is lost and schedules a re-transmission.

If medium is sensed to be busy when planning to transmit a data or ACK frame,

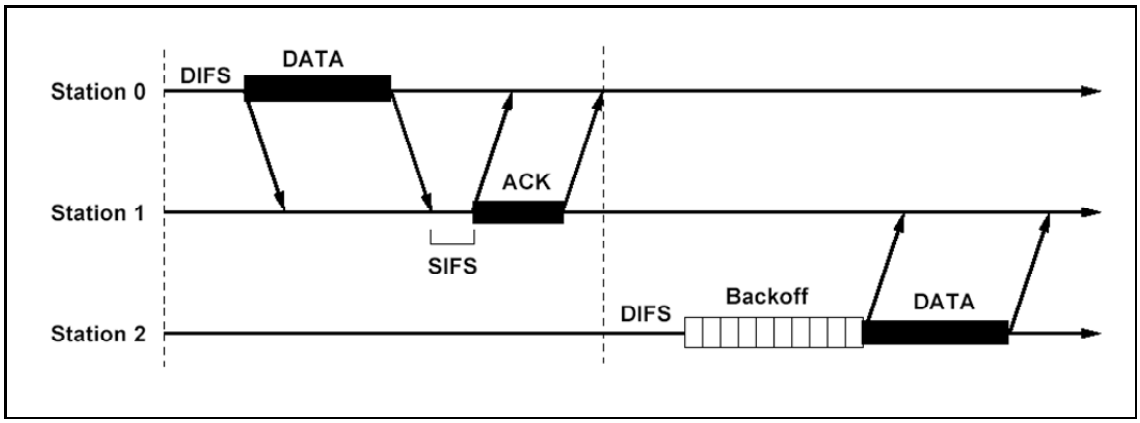


Figure 2.3. IEEE 802.11 MAC basic access method

the transmission is deferred until the end of the current transmission. In scheduling a re-transmission, a random back-off interval is to be selected. For this reason, a back-off timer is constructed by selecting a back-off integer (BV) randomly from a uniform distribution over the interval $[0, CW - 1]$, where CW (Collision Window) is an integer within the range $[CW_{min}, CW_{max}]$. The selected integer BV is the number of idle slots the station must wait before starting transmission (the clear slot definition is found in [17]). The BV value is decremented by one for each detected idle slot. If another transmission starts before BV reaches to zero, decrementing stops and resumes only after this transmission finishes (meaning the channel is idle for DIFS time). The station starts transmitting its frame when the value reaches zero. For each successive re-transmission, the value of CW is set to $CW * 2 - 1$ to introduce an exponential increase. CW is increased in this way until it reaches to CW_{max} and increased no more. When a successful transmission occurs, CW is reset to CW_{min}. This method is proposed to provide minimum collisions for varying utilizations of the network. An illustration to clarify this process is given in Figure 2.3.

DCF has also another access strategy to provide transmission of data frames. This method facilitates the use of RTS/CTS mechanism as mentioned in Section 2.2.1. RTS/CTS messages are used to reserve channel for the upcoming transmission. RTS message is transmitted by the station which has a data frame to transmit and the intended receiver of the frame responds with a CTS message. For RTS and CTS transmissions, the channel is checked as in the ACK frame transmission. RTS and CTS

frames carry the time period during which the upcoming transmission will take place. Upon receiving this information, all neighboring nodes construct a channel reservation table, called NAV (Network Allocation Vector). This method of channel management is known as virtual carrier sense mechanism, and helps reducing contentions due to hidden terminals.

It is vital to understand the offered load versus throughput characteristic of IEEE 802.11 under varying number of stations because CSMA type channel access starts to have high impact on the upper layer protocol performance. In [21], Liu *et al.* show that as offered load to the network increases from 60 to 80 percent, the packet loss, fraction of retransmissions, and packet service times all increase dramatically .

2.3. Mobility Models

Mobility models are widely used for hosts of the ad hoc networks in order to analyze and simulate the physical behavior of mobile nodes in the network under consideration. Performance of the protocol that is to be inspected is highly dependent on the traces of movement that represent characteristics of users of the network. It is possible to use pre-recorded movements of mobiles that have been gathered for a similar scenario. However, it is usually costly to collect such real world traces and finding a representative scenario is not always a trivial task. For this reason, artificially generated movements of mobiles are used in ad hoc network studies. The models for this purpose may be broadly put into two categories as *independent node mobility models* and *group mobility models*. A more detailed survey of mobility models for ad hoc network research can be found in [22].

2.3.1. Independent Node Mobility Models

Models that belong to this class generate mobility traces of hosts independent of each other. Many such models have been proposed in the literature and some of the commonly used models are mentioned with their key characteristics in this subsection.

Random Walk Mobility Model is a simplistic approach to mobility pattern generation, where mobile nodes move to their next location with a randomly selected speed and direction. After a pre-determined constant amount of time or distance, mobile node selects another random speed and direction to travel, completely independent of its previous move. Nodes that reach the terrain dimensions are usually bounced back or re-forced into the terrain from the mirror reflection of its exit point.

Random Waypoint Mobility Model [23] is an extended random mobility model where nodes stay at their locations for a determined amount of time, called *pause time*. Between pauses, node chooses a random destination and starts to travel to that destination with a speed distributed between $[v_{min}, v_{max}]$. Upon arrival, another pause takes place. This model is also used in our simulation experiments for the proposed architecture (see Section 6.4).

Random Direction Mobility Model [24] is an enhancement made to random waypoint mobility model to eliminate the effect of convergence. In random waypoint mobility model, nodes are more likely to travel using the middle of the simulation area in order to reach their next destinations. To overcome this effect, nodes do not stop travelling until they reach the terrain boundary on their selected direction. Upon reaching the terrain boundary, node then pauses and initiates another movement.

Boundless Simulation Area Mobility Model [25] provides a previous move dependent velocity and direction for a mobile node's movement. At every time step, Δt, velocity of the mobile is incremented or decremented Δv amount. Δv is uniformly distributed between $[0, A_{max} * \Delta t]$, where A_{max} is the maximum possible acceleration for a mobile node. If velocity is computed below zero or above a v_{max}, it is assumed zero or v_{max} respectively. Similarly the new direction of mobile is selected by incrementing or decrementing a $\Delta\theta$ amount of change to current direction. $\Delta\theta$ is uniformly distributed between $[0, \alpha_{max} * \Delta t]$, where α_{max} is the maximum possible angular change in direction. In this model, nodes may well travel without bouncing on the terrain boundaries, instead they re-appear on the opposite side of the terrain.

Markov Mobility Models constitute a group of mobility models, that have been used for cellular and ad hoc network mobility modeling. In two-dimensional markov model, probability of moving left, moving right, and staying stationary are defined. Similar approaches are used for other types of markov models (e.g., probabilities of moving to any of the six neighboring cells are defined for some cellular networks).

Smooth Random Mobility Model [26] is another extension of the simpler random walk model where two independent stochastic processes are used to trigger direction and speed changes. The new speeds, for example, are chosen from a weighted distribution of preferred speeds. Upon such a trigger, the speed (or direction) changes as determined by a Poisson process.

Random Gauss-Markov Mobility Model [27] is an improvement over the Smooth Random Mobility Model. In the model, new speed and direction of the mobile is determined by using its past speed and direction plus a random variable. The randomness in the model is provided by this *tuning parameter*. This model eliminates the sudden and sharp movements seen in the Random Walk Mobility Model.

Also in [28], Tuğcu and Ersoy present a *realistic mobility model*, where different classes of mobiles are distributed and driven to move on a real city district map. Mobiles behave according to the class they belong to (i.e., vehicular mobiles travel on highways with high speeds or low speed mobiles walk on streets). Also transitions between different mobile classes are provided to reflect the realistic behavior of *real world mobiles* (i.e., people getting in their cars, travelling on highways to reach home, walk to their houses and stand still).

2.3.2. Group Mobility Models

Mobility models fall into this category because generated mobile node movements are dependent on each other. Whatever the underlying motivation is, nodes move in a somehow cooperative manner (e.g., attacking an enemy, sightseeing a tourist attraction or rescuing people). Mobility models, for this reason, should reflect this behavior in

the generated traces. This subsection briefly mentions some of the well know group mobility models. Compared to independent ones, group mobility models are rather new and less studied.

The basic form of a group motion in mobility models appear in [29] as Fluid Flow Mobility Model. The motion of nodes are modelled as a set of constant velocity fluid flow equations. However, the concept of fluid flow is used in defining mobility of nodes of a cellular network (i.e., traversing cells) in the given reference.

Column Mobility, Pursue Mobility and Nomadic Community Mobility Models are described in [30] and given in intuitive explanations. The three mobility models may be referred to as a less general form of Reference Group Mobility Model (RPGM) [31]. In RPGM, random motions of mobile node groups and random motions of mobiles within those groups are defined. Each group has a logical center, around which the movement of the overall group is constructed. Nodes in Column Mobility Model are placed on an initial grid and each node may move within a constant distance from its point on grid. As the grid is moved along, nodes follow the movement of the grid, generating an overall group movement. In Pursue Mobility Model, a selected node within the group is tracked by the rest of the group. This behavior generates mobility traces for a target chasing environment. Nomadic Community Mobility Model represents a group of mobiles that travel according to a reference point and use their independent mobility models to roam around the given reference point.

3. STATE OF THE ART IN AD HOC NETWORKS

Ad hoc networks offer new applications in many areas together with their important technical challenges. Many researchers attack different problems of ad hoc networking for years. Among those research topics, some of them are frequently addressed like routing and MAC layer issues. Table 3.1 is simplified from the categorization in [32] and classifies the published ad hoc networking related research in year 2001 within the IEEE organization. This classification gives an overview of active research areas on ad hoc networks. However, this is not the only way to classify the available work. For example, among those publications, twenty-two of them present a *power-aware* scheme for ad hoc networks.

To provide the reader with necessary background on related research, following two sections classify and examine some of the available work into two groups *routing* and *service discovery*. The work appearing on this chapter is by no means exhaustive and only gathers some of the ad hoc networking research that we believed to be interrelated with the proposed work in this thesis.

3.1. Routing

The multi-hop ad hoc network routing problem is one of the first and very frequently attacked problem in the history of ad hoc networks. Relaying packets that belong to someone else is an inherent obligation from the use of limited range wireless communication equipment. Since routing protocols for wired and infrastructured networks have been extensively studied [4], many of the proposed ad hoc routing algorithms have roots in those well-known approaches. Traditionally, the routing protocols are evaluated in two main categories: proactive and reactive. A survey of available ad hoc network routing protocols is given in both [12] and [33].

Table 3.1. Classification of ad hoc networking research in year 2001

Research Category	**Number of Publications**
Routing	73
MAC, Scheduling, Physical layer	31
Special Ad Hoc Networks	18
Applications	17
Clustering, Organization, Topology	16
General Overviews	9
Internet Protocols on Ad Hoc Networks	8
Network Management	7
QoS, Service Differentiation	7
New Network Concepts	5
Service Availability	5
Positioning, Situation Awareness	5
Transport Issues	2
Security	2
Mobility	1

3.1.1. Proactive Routing Protocols

Proactive routing protocols try to discover the topology of the ad hoc network by exchanging topological information among nodes. The discovered topological information is stored in tables for future use and these kind of routing approaches are sometimes referred to as *table-driven protocols*. Any node on the network has the advantage of finding a route to the destination node immediately. Price for this information is paid in the extra control traffic generated.

First proposed ad hoc routing algorithms were proactive type and usually based on *Distributed Bellman-Ford* (DBF) algorithm [34]. Attacking excessive control traffic and convergence problems of DBF, many variants are found in the literature such as [35] and [36].

Destination Sequenced Distance-Vector Routing(DSDV) algorithm is a well known

improvement on DBF, appeared in [37]. DSDV introduces full table broadcast and event-driven incremental updates to limit control message overhead. Table updates are emitted with sequence numbers to ensure fresh route information at nodes. Also DSDV lets nodes to settle before they send table information to neighbors. *Global State Routing* (GSR) [38] is similar to DSDV and takes the idea of link state routing but improves it by avoiding flooding of routing messages. *Fisheye State Routing* (FSR) [39] is an adaptation of GSR, where a given node exchanges topology information about closer nodes more frequently than it does about farther nodes. This reduces the update message size used in GSR. Introducing such hierarchy into routing is frequently used. *Hierarchical State Routing* (HSR), as presented in [40], is a multilevel clustering and logical partitioning of mobile nodes. Each node has a hierarchical address and routing is realized via cluster-heads. *Clusterhead Gateway Switch Routing* (CGSR) [41] also uses DSDV as a basis and constructs the hierarchy by selecting cluster-heads and gateway nodes. This way, routing is simplified into a process of packet transmission to cluster-heads. Cluster-heads know what to do (forwarding to necessary gateway) with the packet.

A different approach in proactive routing protocols is to apply link state protocols to the ad hoc network routing problem. A good example of this approach is the *Optimized Link State Routing* (OLSR) algorithm [42]. In principle, OLSR specifies link state algorithms on *multi point relay* (MPR) set of nodes. A node‘s MPR set is a subset of its neighbors whose combined radio range covers all nodes two hops away. Routes are computed and update information is forwarded using a node's partial view of the network. *Zone-based Hierarchical Link State Routing* (ZHLS) [43] is a link state protocol variant and introduces hierarchy to provide virtual links between non-overlapping partitions, called *zones*. Routing is realized by making use of zone id values, till packets reach the correct zone. Then node id values are used within zones. *Core-Extraction Distributed Ad hoc Routing* (CEDAR) [44] is another link state protocol variant where routing decision is made based on QoS. The virtual backbone is termed as core and is basically computed using the *Minimum Dominating Set* scheme. This is done by maintaining local states and executing local computations. Propagating the bandwidth information across the virtual backbone performs QoS routing. The route

from the source to the destination is computed taking into account the maximum available bandwidth path with minimum number of hops.

There are also proposals that combine features of distance vector and link state approaches. *Wireless Routing Protocol* (WRP) [45] is such an example where each node in the network maintains a distance table, a routing table, a link-cost table and a message retransmission list. Compared to DBF, it reduces the amount of route looping and ensures reliable update message exchange.

3.1.2. Reactive Routing Protocols

Reactive routing protocols, which are often called *on demand* routing methods, are based on the idea of *finding routes as needed*. This idea eliminates the overhead of proactive approach where control messages are exchanged in the network to maintain up-to-date topological information. However, route discovery latency is of concern for reactive approaches, since finding route on the fly requires flooded query messages on the network.

Ad hoc On-demand Distance Vector Routing (AODV) [46] is the *on demand* translation of DSDV (See Section 3.1.1). Each AODV running node keeps a destination indexed next-hop routing table. If a route is needed for an unseen destination, a route request message is broadcast (using an expanding ring). As the route request message travels, it updates the tables of hosts on route (about the source node). A unicast packet is replied back either from an immediate node (which knows the way to the destination) or from destination itself. This way, source discovers the route on the fly. On broken link failures, source (and the hosts on the broken route) is informed and a new route discovery is initiated if necessary.

Dynamic Source Routing (DSR) [23] is an on demand source routing protocol. A DSR running node maintains route caches containing the source routes that it is aware of. The node updates entries in the route cache as and when it learns about new routes. If a source has no recorded route to destination, it initiates a route discovery by

broadcasting a route request packet to its neighbors. The route request packet contains the address of the source and the destination, and a unique identification number. Each intermediate node checks whether it knows of a route to the destination. If it does not, it appends its address to the route record of the packet and forwards the packet to its neighbors. To limit the number of route requests propagated, a node processes the route request packet only if it has not already seen the packet and its address is not present in the route record of the packet. A route reply is generated when either the destination or an intermediate node with current information about the destination receives the route request packet. A route request packet reaching such a node already contains, in its route record, the sequence of hops taken from the source to this node. On broken link failures, source is informed with a route error packet and a new route discovery is initiated if necessary. There are many optimizations specified for DSR that increases its efficiency. These are given in [47].

The Temporally Ordered Routing Algorithm (TORA) [48] is an on demand protocol designed to minimize reaction to topological changes. A key concept in its design is that it decouples the generation of potentially far-reaching control message propagation from the rate of topological changes. Such messaging is typically localized to a very small set of nodes near the change without having to resort to a dynamic, hierarchical routing solution with its attendant complexity. In TORA, routes are established only when necessary by constructing a directed acyclic graph rooted at the destination using a *query and reply* process. Reaction to link failure is only given when necessary (i.e., when a node loses its last downstream link) and scope of failure reactions are minimized (i.e., the number of nodes that must participate).

Associativity Based Routing (ABR) [49] is based on the concept of associativity, and new routing metrics are introduced, which are the longevity of a route (route relaying load) and the link capacity. The basic idea of associativity is that there is no point in choosing a shortest-hop route if a route is going to be invalidated due to the node's mobility. Every node learns its 'association' with the surrounding nodes, where association can mean signal strength, power life, period of presence or spatial and temporal characteristics. The key idea is to choose a route that goes through nodes

having a high degree of association stability. ABR has a similar route discovery phase.

Signal Stability-Based Adaptive Routing (SSR) [50] is another on-demand routing protocol that selects routes based on the signal strength between nodes and a node's location stability. This route selection criterion has the effect of choosing routes that have 'stronger' connectivity.

Location Aided Routing (LAR) [51] is an on-demand protocol that makes use of physical location information of destination node to reduce the search space for route discovery. Instead of flooding the whole network with route discovery message, this protocol send messages to a subset of nodes from whom the probability of finding route is very high. After a few unsuccessful attempts, this subset may grow to the size of whole network. Thus it attempts to reduce the latency of route determination.

Also, gossiping (originally proposed for probabilistic multicast) has recently been applied to minimize query traffic while flooding in reactive on demand routing protocols. In [52], authors propose a gossiping-based approach, where each node forwards a message with some probability, to reduce the overhead of the routing protocols.

3.1.3. Hybrid Routing Protocols

Proactive routing protocols can generate the required route very quickly, but waste too much of the available bandwidth for network topology exchange information. Reactive protocols may reduce this waste of bandwidth but they may encounter excessive delays in finding a route because of flooding the network with route queries. A middle-way approach is to incorporate features from both 'extremes'. This approach is seen in *hybrid routing protocols*.

Zone Routing Protocol (ZRP) [53] is such a hybrid approach where route determination is on-demand, but with limited cost of the global search. In this approach, a *Routing Zone* is defined for each node which includes the nodes whose distance is at most some predefined number (e.g., in terms of number of hops). This distance is

referred to as the *zone radius*. Each node is required to know the topology of the network within its zone only and any updates about change in topology are done within the zone only. That is updates are only locally propagated.

Distributed Dynamic Routing Algorithm (DDR) [54] is a hierarchical hybrid routing algorithm which is an improvement over ZRP. It greatly reduces routing complexity and increases delay performance. The basic idea is to construct a forest from a network topology, which consists of connected non-overlapping dynamic zones (also called trees).

3.2. Service Discovery

With the increase of the available services that are accessible via computer networks, providing ways to access those services gained more importance. The 'waiting goal' to achieve is to provide users with such a framework that no user configuration of any kind is necessary to find, locate, and use the available services on the network. For this purpose, service discovery protocols provide the mechanisms for users to:

- Lookup and browse around the available services
- Extract necessary information to be able to select the right service
- Utilize the service

Role of service discovery is significant when ad hoc networking is of concern because there exists no infrastructure inherited from the network itself. For example, using a service discovery protocol, a notebook capable of using Bluetooth technology, may be able to discover the GPRS capability (GPRS connection service) of a nearby cellular phone, in order to get access to Internet for new e-mail check. In the following subsections, currently available service discovery protocols are summarized.

3.2.1. Service Location Protocol

Service Location Protocol (SLP) [55] is being developed by IETF and aims to be a vendor-neutral standard supporting 'spontaneous' discovery of services. In the defined SLP architecture, three elements exist:

- **User Agents (UA)** : Realize service discovery on behalf of clients
- **Service Agents (SA)** : Advertise location and attributes of services on behalf of service offering applications
- **Directory Agents (DA)** : Gather service location and attribute information from SAs and record them into a database for future references (to respond queries)

Whenever a new service is offered, the SA contacts the DA to advertise this new service. If a user needs to find a certain service, it queries the available services in the DA. Only after getting location and attributes of the service, client may start utilizing the service.

There may be more than one DA in the network, which actually increases the efficiency of the protocol because SA and UA uses unicast messages to communicate with DAs. Before SAs and UAs try to talk to a DA, they first need to obtain address of it via DHCP.

Services are defined by using Service URLs and Service Templates. They encode the address, type and attributes of the service. A sample service URL and its associated template is given in Figure 3.1.

Currently, SLP version 2 is available, and version 1 has been implemented in commercial products supporting HP JetSend technology (like printers, digital cameras, scanners, PDAs etc.).

```
service:printer//lj4050.tum.deL1020/queue
scopes = tum, bmw, administrator
printer-location = Room 0409
pages-per-minute = 9
printer-name = lj4050
```

Figure 3.1. A sample service URL and its associated template for SLP

3.2.2. JINI

JINI is a Java environment extension and developed by Sun Microsystems Inc. [56]. Since SLP has its roots at Sun Microsystems, there are some inherit similarities in both protocols. However, JINI is tightly coupled to Java, where SLP is provided as vendor-neutral.

JINI addresses the issue of how devices connect to each other in order to form a simple ad hoc network, named *a JINI community*. For this purpose, JINI provides an architecture and a programming model.

All hosts that will use JINI are assumed to have a working Java Virtual Machine (JVM) implementation. Service registration process of JINI is called *discovery* and *join*. Similar to DA in SLP, JINI has *Lookup Table* on a lookup server, where services on the network are recorded. Different from DA entries, JINI can store Java codes to Lookup Table. This feature enables the service clients to download device drivers or interfaces from the Lookup Table as well.

The JINI specifications and reference implementations are open source and may be freely downloaded for non-commercial use from Sun Microsystems web site [57].

3.2.3. Salutation

Salutation is an approach to discover services and developed by an open industry consortium, called *Salutation Consortium* (composed of companies including HP,

IBM, Xerox and AOL) [58]. The architecture of Salutation includes three components: Client, Server and Salutation Manager (SLM).

SLM manages all communication and sometimes referred to as service broker. Services register themselves to SLM and clients query SLM when they need a service. After discovery, clients are able to request utilization of the service from SLM. All Salutation protocol is based on SunRPC, which seems to be the only technology dependent part of the design. SunRPC provides no multicast other than broadcast and does not provide strong encryption.

Salutation may operate without a directory, where clients and services locate each other directly. This allows a flexible protocol, where establishing a directory service is costly (in automobiles, home etc.). Salutation is designed to be highly compatible with available wireless technologies, and already have commercial implementations.

3.2.4. Universal Plug and Play

Universal Plug and Play (UPnP) is a standard for spontaneous configuration developed by *The Universal Plug and Play Forum*, which is led by Microsoft Corp. [59]. UPnP extends Microsoft Plug and Play technology to devices that are accessible via a computer network. As an architecture, UPnP resembles Salutation and SLP. It uses XML for device definitions, service definitions and queries on them. UPnP requires IP, HTTP and XML support to function correctly.

A UPnP device is said to export one or more services and the service definition XML can be complete abstract description of the type of service, the interface to a specific instance of the service and even be the ongoing state of the service.

There is no central service directory to which services register themselves. The Simple Service Discovery Protocol (SSDP) [60] is used within UPnP to find and locate services. Starting from the URL provided by the definition XML, SSDP defines a web based discovery protocol, which uses HTTP.

3.2.5. Bluetooth Service Discovery Protocol

Bluetooth is a new short range wireless transmission technology containing *Bluetooth Service Discovery Protocol* (SDP) used to locate the services provided by or available via Bluetooth devices. Details of SDP is given in [61] and modified from the Piano platform of Motorola to suit dynamic nature of ad hoc communications.

SDP supports search for services by type and by attribute. Also SDP allows users to browse available services without any prior knowledge of service characteristics. SDP has no functionality in accessing services, and once discovered, selecting, accessing and utilizing is done with mechanisms out of the scope of SDP (e.g., may be realized by using other service discovery protocols like SLP or Salutation).

3.2.6. Secure Service Discovery Service

The *Secure Service Discovery Service* (SSDS) is a University of California, Berkeley research project [62]. SDSS is quite similar to other discovery methods with improvements in reliability, scalability and security. SSDS is implemented in Java but unlike JINI, uses XML for service description and location.

SSDS architecture has clients, services and Secure Discovery Service (SDS) Servers. SDS Service availability is periodically announced by multicast messages containing URLs for the available SDS server. Architecture provides ways to add more servers to scale up the system under heavy load.

The key characteristic of SSDS is security and in the architecture, all parties are authenticated before operations. Furthermore, all message traffic is encrypted.

3.2.7. Intentional Naming System

Intentional Naming System (INS) [63] is designed to discover resources and locate services for dynamic and mobile networks of devices and computers. INS applications

may be services or clients. Clients request and access the data or functionality provided by services.

Intentional Name Resolvers (INR) form an application-level overlay network to exchange descriptions of services, construct a local cache based on these advertisements and route client requests to the appropriate services. INRs are self configured with no manual intervention. INS uses a simple descriptive language with attribute-value pairs for its names, which enable the service to be specified precisely.

INS has a late binding mechanism that integrates name resolution and message routing. This enables clients to continue communicating with end nodes even if the name-to-address mappings change while a session is in progress. INS applications may submit data to the service together with the definition of the required service. Therefore INRs play important role in binding and routing.

In order INS to scale large number of names and services, *Domain Space Resolvers* (DSRs) and *partitioned virtual spaces* are defined to restrict the working scope of INRs.

4. PROPOSED PROTOCOL

SeMA is an architectural approach to mobile wireless ad hoc networks (MANET) with the fundamental motivation of defining a complete ad hoc network in the sense that applications are provided with a framework of protocols and primitive mechanisms. SeMA is an acronym for 'Session Based Multi-layer Ad Hoc Network Architecture' and has also been interchangeably used for 'Service Aware Mobile Ad Hoc Network Architecture'. Basics of the architecture have first appeared in [64].

The protocol proposal in this thesis forms a part of the SeMA architecture, trying to provide mechanisms to define, announce and access to services for ad hoc network hosts. While realizing this, issues of routing, session management and binding are addressed for a complete discussion of an ad hoc network architecture. Unless otherwise noted, in the context of this thesis, the acronym SeMA is frequently used to refer to the proposed protocol itself, discarding the rest of the architecture elements.

4.1. Overview

SeMA provides a protocol stack operating with three components to accommodate access to general purpose network applications such as printing, file transfer and so on in a highly mobile dynamic ad hoc environment. In such a target environment, neither an infrastructured network underneath nor an overlay lookup network exist. Nodes are assumed to be willing to relay each other in order to maximize their access to each others services.

The ad hoc network is formed, maintained, and terminated for the simple purpose of benefiting from a service provided by some other host of the network. The lifetime of the network is therefore determined by the duration of the client server interaction. In this sense, SeMA defines the ad hoc network to be a session (either transparent to the application or not) between communicating parties. According to this definition, a group of wireless communication capable mobile nodes do not necessarily form an ad

hoc network. An activity of communication for service access (a session) is necessary to call them a part of an ad hoc network. IETF MANET working group's definition of an ad hoc network in Chapter 2, however, does not mention such a property.

Services are announced in the SeMA network for possible future uses by the ad hoc network hosts. But the service binding is delayed until the actual service request is made by the application. SeMA also provides mechanisms to discover the services whose announcements have not reached before the application request is made.

For the operation of proposed protocol, a wireless medium access control (MAC) and link layer is required. Only broadcast and unicast frame transmission and link failure detection functionality are assumed to be supported in the data link layer. Broadcast frame transmission capability is found in almost all available data link layer protocols because of the broadcast nature of the wireless medium. This capability is necessary to disseminate announcement type information through the SeMA network. Link failure detection facility is required to sense and act accordingly (i.e., triggering some protocol algorithms) in case of a unicast packet transmission failure. This feature also exists in many of the available wireless data link layer protocols, including popular IEEE 802.11 variants.

SeMA architecture supports network layer addressing without any need for newly introduced unique identifiers for mobile nodes (such as IP addresses) and make use of already available data link layer addresses as identifiers. Apart from being unique addresses, classical network layer addressing provides no extra information or ease for protocols so far as our protocol proposal is concerned. A popularly used approach, (in almost all available literature as the time of writing) is to use IP as the network layer and develop algorithms of routing, service discovery etc. at IP layer. Although using IP provides provision for native access to other IP networks (i.e., fixed infrastructure) and introduces ability to connect inhomogeneous media, motivation of *network address* and *host address* approach (and its related IP routing algorithms) of IP does not make sense for an ad hoc network where no infrastructure exists at any level. SeMA applications use a slim protocol stack without any IP mechanisms embedded inherently.

For our proposal, possible services on ad hoc networks are classified into two categories, namely *non-interactive* and *interactive*. Non-interactive services are kind of services that usually require best effort delivery and contain no constant interaction of client and server. This kind of service is usually for applications that only need to deliver its intended data to remote party. A *printing service* can be given as a non-interactive service type example. Printing service client submits its document for printing, and only further interaction is the reception of the success indication for the submitted job from the printer service provider host. Interactive services, however, requires interaction from remote endpoint of communication and have applications like file transfer, peer-to-peer voice and video communication. These kind of services require more than best effort delivery (i.e., reliable delivery for a file transfer application, or delay bound for a video application) and SeMA layer introduces extra mechanisms to satisfy those requirements (e.g., network layer acknowledgements). To clarify the type of service provided by the protocol, non-interactive service support can be classified into 'connectionless', interactive service support can be classified into 'connection oriented' approach of communication. The rest of the document specifies the protocol, discusses and presents the results that are related to non-interactive services, which has been studied extensively for the course of the thesis.

Services in the proposed ad hoc network are discovered to start a session, and a valid source route is extracted from the announced service. Then packets of this session are forwarded through the relaying hosts according to this initial route. Route optimizations take place at all intermediate nodes to overcome link breaks, mainly because of the mobility of those nodes.

We have already defined the ad hoc network as a session between the service provider and its user. Keeping this in mind, sessions of non-interactive services are maintained by SeMA, transparent to the user. These transparent sessions start with the late binding to the service of concern (after the discovery and lookup phases if necessary) and end with the termination indication from the remote party. Service client is not aware of the internals of the ongoing session, and only presented with a success indicator if all packets of the session are delivered to the service owner host.

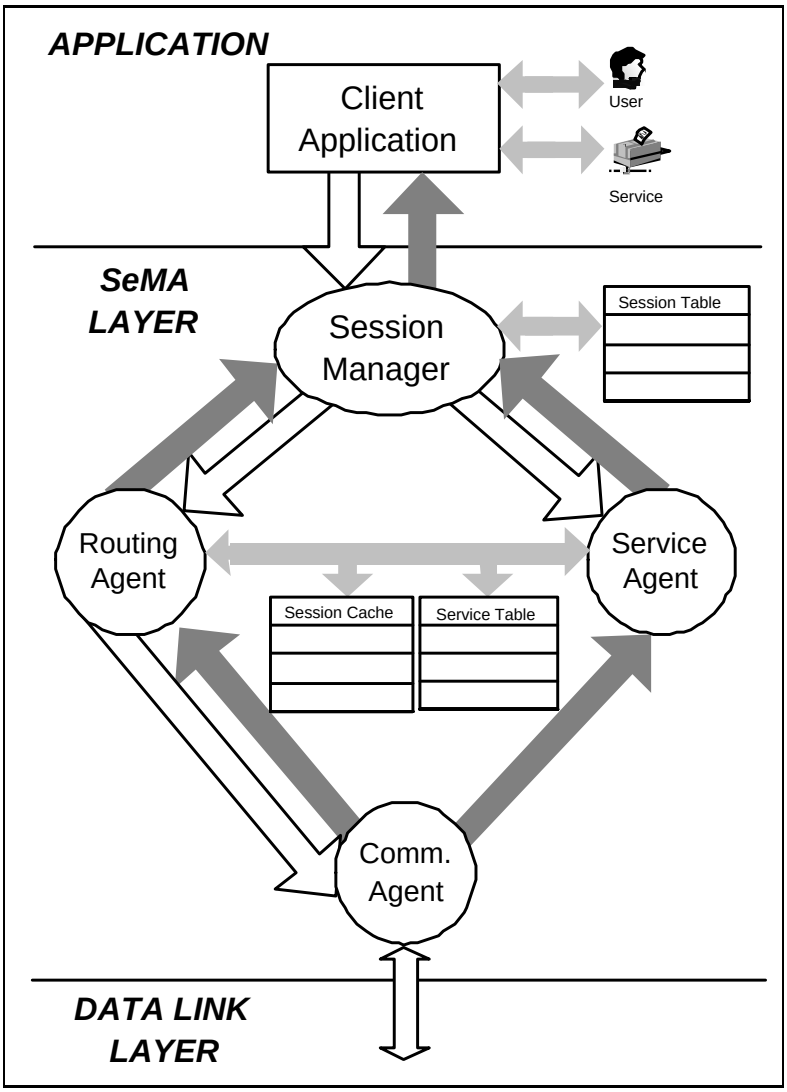


Figure 4.1. An overall look to the SeMA protocol stack

Session maintained by SeMA for interactive services, however, is not transparent to the application and application explicitly uses the provided session handle to further maintain and send data over the session. Application also voluntarily terminates the session, which was done implicitly *on delivery* for non-interactive services.

The overall look to the components and their interactions for the proposed protocol is illustrated in Figure 4.1. The Section 4.4 provides a closer look to the individual components of the overall picture.

4.2. Modelling of Protocol Elements

For the proposed ad hoc networking protocol, it is vital to clearly define and represent elements of the network model under consideration. These representations are to be used both in formal definitions of protocols and in the implementation of protocol algorithms. Details and motivations behind this representation has also appeared in [65].

The two key elements of the protocol are host and service instances. These instances are defined and represented as extensible markup language (*XML*) [66] instances that conform to an *XML Schema* [67], designed to specify structure and content of the instances. Before processing an XML instance, the document is first checked to be well-formed according to the XML version used. Then, the document is validated using its respective schema document, in order to ensure that the rules specified in the schema are honored in the instance.

XML is chosen since it provides a flexible, easy-to-parse, structured text-only format to access data efficiently. However, a practical disadvantage of using a human readable text format is the size of the XML instances generated. Some of the instances are to be carried in protocol data packets and text-only format is not effective for this purpose. Details and a solution for this problem are given in Section 5.2.1. Following two sections discuss how host and service instances are defined in the protocol.

4.2.1. Modelling Hosts

Hosts are defined to conform the XML schema document named *hostSchema* and have various tags and attributes. Listing of the *hostSchema* XML schema document (XSD) that is required to validate mobile host XML instances is given in Appendix A. Although all existing attributes of a host instance are related to that host, some of them are not needed by the protocol itself but used by the simulation software of the protocol. Current coordinates of the host, for example, is homed by its XML instance but not required by protocol mechanisms to function properly.

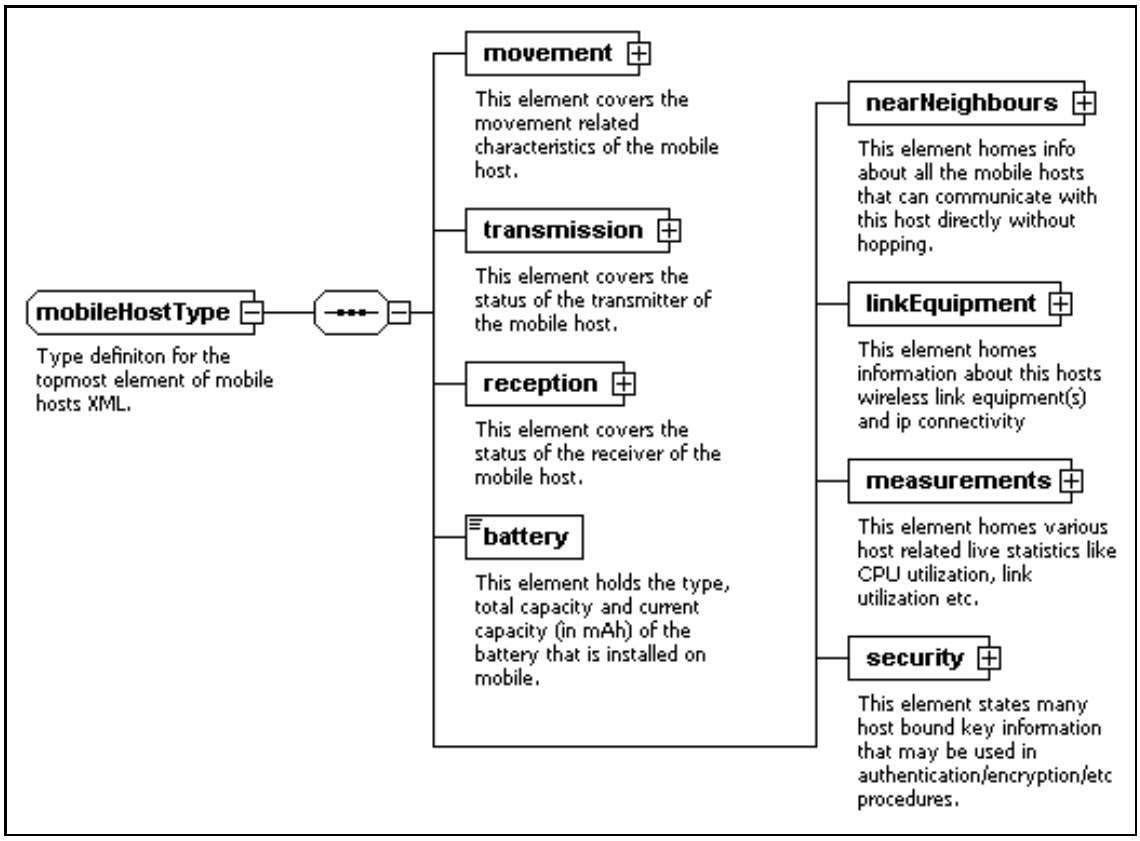


Figure 4.2. Main and first level child elements of host XML documents

Feature-based characteristic of the host instances helps to build protocols that are aware of a specific feature (e.g., a power-aware routing scheme or secure service discovery). The main element and first level child elements of the host XML documents are illustrated in Figure 4.2.

Following discusses some of the features of mobile host XML instances using the sample XML instance provided in Figure 4.3 on Page 39. Mobile hosts are identified by their wireless link layer addresses, which is discussed to be necessary and sufficient in Section 4.1. This attribute is in the main *mobileHost* tag and is the only mandatory host XML instance field. The *movement*, *transmission*, and *reception* are simulation-related tags and their attributes help simulation software to properly locate, move, and determine nodes and their wireless communication ranges (not all of them are currently used in developed simulation software). The *linkEquipment* tag homes host's wireless and wired interface connection properties, such as link interface card type (e.g., IEEE 802.11b) and brand. If host has access to an IP network, its IP address is also given in

```
<mobileHost linkID="A3:55:4F:C3:64:B4" lastUpdate="2003-01-22T13:20:00.000-05:00">
  <movement>
    <currentMove x="13" y="44" speedunit="m/s" speed="244">In Region 4</currentMove>
    <nextMove speedunit="m/s" speed="200">45</nextMove>
  </movement>
  <transmission>
    <currentPower>200</currentPower>
    <maxRange>500</maxRange>
    <shape>circle</shape>
  </transmission>
  <reception>
    <currentPower>120</currentPower>
    <maxRange>500</maxRange>
    <shape>circle</shape>
  </reception>
  <linkEquipment>
    <bluetooth id="7">disabled</bluetooth>
    <ieee802.11 ver="a">cisco aironet</ieee802.11>
    <ieee802.11 ver="g">proxim</ieee802.11>
    <ip connectivity="yes">193.244.55.61</ip>
  </linkEquipment>
  <measurements>
    <cpu utilization_avg="67">i386</cpu>
    <wlink utilization_avg="11" collision_avg="0.01"/>
  </measurements>
  <battery type="NiMh" maxCapacity="450">200</battery>
  <security level="high">
    <publicKey id="03" algorithmName="PGP" length="64">110...00</publicKey>
  </security>
</mobileHost>
```

Figure 4.3. XML definition of a sample mobile host

this tag. The *measurements* tag has live measurements of some host related attributes that are updated throughout the lifetime of the host. The *battery* is a separate tag in a mobile host instance since battery related features of a mobile are usually very important and vital. Many protocol features and behavior of applications can be related to battery type and capacity of the mobile hosts under consideration. The *security* tag is suitable to announce mobile host's abilities of cryptographic communication, including its public keys and algorithms.

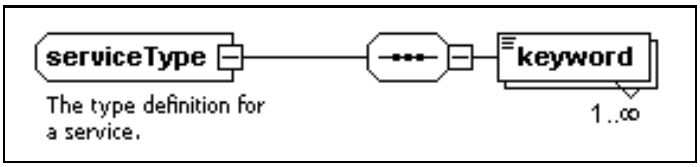


Figure 4.4. Main and first level child elements of service XML documents

4.2.2. Modelling Services

All available resources offered for use by mobile hosts are formally defined as services of the ad hoc network under consideration. Services constitute the backbone of the protocol as many of the mechanisms are built with services provided by mobile hosts of the ad hoc network in mind. The overall idea is such that hosts of the SeMA architecture try to lookup, discover and bind to services offered in the network. They do not try to find ways to reach directly to hosts and then services.

The specification of a service should include necessary information for an ad hoc network host to correctly determine whether its need for the service can be fulfilled by this instance. A similar descriptive modelling approach is used for *Service Location Protocol* (SLP), and Guttman *et al.* give service types in *attribute* and *value* form in [55]. Similarly in our protocol, service instances are represented as attribute-value pairs where an attribute is a category in which a service can be evaluated (e.g., *papersize* for a printer). A value is the classification of the service within that category (e.g., *A4* for *papersize* attribute). Attribute and values are free form strings defined by applications that offer those instances, and may only be meaningful for those client-server application pairs. But names of services are to be organized in a coordinated manner since at least a name is required to identify a service instance. This coordination may be realized by a naming authority, similar to what *Internet Assigned Numbers Authority* (IANA) does for SLP. This simple element structure of *serviceSchema* document to which service XML documents are supposed to conform is given in Figure 4.4.

In our protocol, mobile hosts announce the services they provide by broadcasting necessary service instance XML documents via the *Service Announcement* mechanism designed. At the service client (i.e., host of the application that will be using this

```
<service name="printer">
    <keyword attribute="location">Engineering B.443</keyword>
    <keyword attribute="color">no</keyword>
    <keyword attribute="papersize">A4</keyword>
    <keyword attribute="papercount">81</keyword>
    <keyword attribute="postscript">yes</keyword>
    <keyword attribute="maxResolution">600*600</keyword>
</service>
```

Figure 4.5. XML definition of a sample printer as a service instance

service), protocol elements do a basic name checking and filtering among the services available to return matching service instances to application. This process is covered in detail in Section 4.4.2. Choosing a service among suitable instances is not a trivial task and it is not the responsibility of the protocol proposed. Features that exist in the available candidate instances are meaningful to application that needs binding. Our protocol tries to provide applications with valid service instances as close as possible to their requests. Selection process among the found instances is the duty of the application. The protocol mechanisms however, has the ability to present some optimal alternatives as long as they refer to the same service instance (e.g., such as offering the shortest hop route alternative to the application).

Figure 4.5 and Figure 4.6 provide two sample service XML instances that is well-formed and valid to be used in SeMA. In Figure 4.5, a *printer* service instance is given with its various attributes and their respective values. For an example scenario using this instance, a SeMA application will not be able to select and bind to this printer if it needs a printer to print a colored document of a hundred pages. Figure 4.6 illustrates how a classical http service may be represented as a SeMA service instance. It will be possible for a mobile host to have classical http service by looking up a service instance named *httpDocument*. IP connectivity of the host that provides this service is another issue and beyond the discussion of the service specifications.

```
<service name="httpDocument">
    <keyword attribute="baseURL">www.netlab.boun.edu.tr/people</keyword>
    <keyword attribute="filename">index.html</keyword>
</service>
```

Figure 4.6. XML definition of a sample web page as a service instance

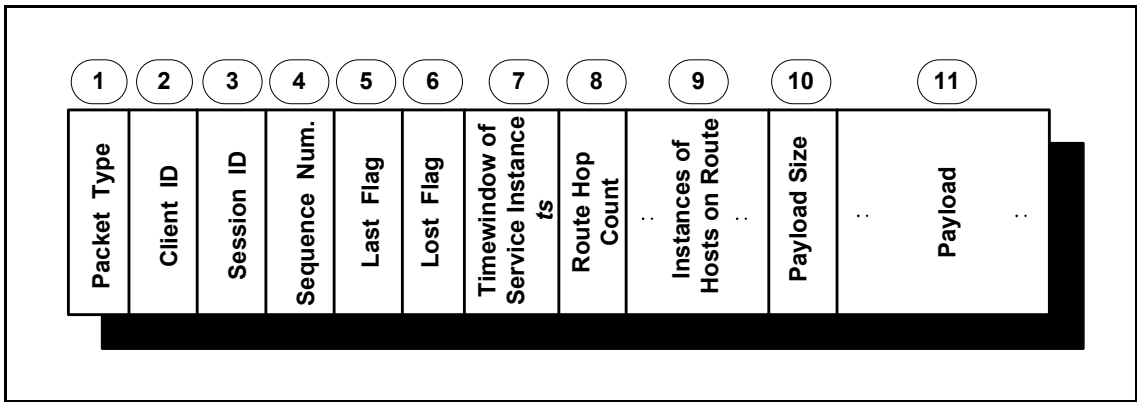


Figure 4.7. SeMA packet structure

4.3. Packet Structure

SeMA protocol stack defines mechanisms to create, maintain and terminate an ad hoc network, and uses the packet structure illustrated in Figure 4.7.

A SeMA packet on the ad hoc network is uniquely identified by the combination of fields (2), (3) and (4) in Figure 4.7. Since there is no fragmentation defined between SeMA and the underlying data link layer, a SeMA packet has to fit into one data link frame, which determines the upper limit on the size of a packet. Maximum available payload space in SeMA packets is determined by the typical reserved space for host instances and expected maximum network diameter (i.e., maximum number of hosts in source routes). The description of fields for the SeMA packet is given as:

1. **Packet Type**: Indicates one of five distinct packet types of SeMA. Their names and brief explanations are given in Table 4.1.
2. **Client Id**: Unique identifier of the node (provided by the data link layer) where the session is originated.
3. **Session Id**: A locally unique identifier generated by the client node for a session which has already, obtained binding. It is 0 when packet is of type *announcement* or *lookup reply*. This field is used as a TTL field when packet is of type *lookup* (See Section 4.4.2.4).
4. **Sequence Number**: A unique, packet identifying number within the session (3)

Table 4.1. SeMA packet types

Packet Type	Explanation
SeMA_DATA	Regular application data for sessions
SeMA_LOOKUP	Broadcast packet to search for services
SeMA_LOOKUPREPLY	Reply to the lookup originator
SeMA_ANNOUNCEMENT	Broadcast packet to advertise services
SeMA_TERMINATE	Session termination indicator

of host (2).

5. **Last Flag**: This flag is used to mark the last packet of the session (See Section 4.4.1.4).
6. **Lost Flag**: This flag is used to mark a packet whose next hop was found unreachable by the last processing routing agent (See Section 4.4.3.2).
7. **Timewindow of Service Instance**: A relative time interval during which the corresponding service is available (See Section 4.4.2.3).
8. **Route Hop Count**: The number of intermediate hosts available in the source route field (9).
9. **Instances of Hosts on Route**: Host XML instances of the intermediate hosts that constitute the source route for this packet. First instance being the originator of the packet, and the last instance being the destination of the packet. These two special instances are not counted in (8).
10. **Payload Size**: Size of the payload field (11) in bytes.
11. **Payload**: The portion of the packet that carries the intended data.

4.4. Components

SeMA protocol is composed of four functional units called *session manager*, *routing agent*, *service agent*, and *communication agent*, as illustrated in Figure 4.1 on Page 36. In the protocol stack, session manager is the unit that interfaces the SeMA layer to applications and communication agent is the unit that interfaces the SeMA layer to the available data link layer. Service and routing agents mainly perform network layer operations, related to packet forwarding and service table maintenance.

Session manager tries to manage the ad hoc network by initiating service fetch process, organizing session packets and terminating the current session.

4.4.1. Session Manager

Session manager coordinates the applications request for service by making protocol primitives available. Its internal mechanisms are built around the idea of the ad hoc network that is maintained for the sake of a session.

4.4.1.1. Announcing Availability of Services. Upon deciding to offer a service available to the hosts of the ad hoc network, an application uses *offer* primitive of the session manager to provide necessary information to the session manager. This information includes a valid and well-formed service XML instance and a validity timestamp. The XML instance is composed of a service name which is a defined word from a broad classification of services, and descriptive attribute-value pairs. With the use of offer primitive, the host becomes the provider of a service, specified by the respective XML instance until the time given in the validity timestamp is reached. The validity time and offering repetitions for a service determine how fresh are the entries found on the hosts of the ad hoc network.

The session manager informs the *service agent* to trigger the necessary action for enabling a service on the network, called *service announcement*. Service announcement is simply a flooded information for the potential ad hoc network hosts stating the availability of a service in the network. Details of service announcement procedure is given in Section 4.4.2.

4.4.1.2. Session Table. *Session table* is the main data structure of the session manager. This table holds information about the *inbound* (originated from remote parties to benefit from services provided by this host) and *outbound* sessions. Some of the fields of a session table entry and their brief usages are given in Table 4.2.

Table 4.2. Important fields of a session table entry

Name of Field	Explanation
sessionType	Indicates the session as either *inbound* or *outbound*
remoteAddr	Host address of the remote party taking role in this session
localSessionId	Locally unique, assigned session id
localApplicationId	Identifier of the local application (provider) taking role in this session
remoteSessionId	Locally unique (at remote side), assigned session id
localApplicationId	Identifier of the local application taking role in this session
departureBuffer	Used to maintain outgoing SeMA packets if session is of type outbound
arrivalBuffer	Used to maintain incoming SeMA packets if session is of type inbound
lastPacketArrived	Indicates the arrival of SeMA packet with last flag set
sessionActive	Used to mark the session as all expected packets arrived, or timeout

4.4.1.3. Maintaining an Inbound Session. An inbound session is started with the first SeMA packet received from the *routing agent*, bearing an unseen remote session identifier from a given remote host. A quick search on the records of the session table is necessary to decide if this packet belongs to an unseen session. If so, a new session table entry is created, it is initialized, a new locally unique session identifier is assigned, and this very first packet of the session is recorded into the sessions *arrival buffer*. The application that offered the service under consideration is not aware of this ongoing session and will be informed of this activity after all packets of the session arrived. With the arrival of the last packet belonging to this session (which may be either the packet with last flag set or an expected out-of-sequence delivery), session manager reorders any out-of-sequence delivery before passing the *pointer to the arrived data* to the application.

Upon the successful reception of all packets, session manager asks routing agent for initiation of a *terminate packet* towards the remote party, which will be the indication of graceful reception of all packets and termination of the ad hoc network. An incomplete session with no new packet arrival in *SEMA_SESSION_TIMEOUT_CHECK* time, which is a protocol parameter, is assumed to be a timed-out session.

Session manager is provided with no duplicate packets for a given session since

routing agent uses a *session cache* to discard previously seen packets. Operation of the session cache is explained in Section 4.4.3.

4.4.1.4. Maintaining an Outbound Session. For an outbound session to start, application first needs to ask for an available service instance that has not expired as the time of binding by using *fetch_service* primitive of session manager. *Service agent* is responsible for *service table* search or *service lookup* procedure which will extract the available matching services (See Section 4.4.2.4 for details). Session manager silently passes found suitable instance alternatives to the application. If application chooses to use one of the alternatives, it passes a pointer to its *data to be submitted* together with the instance identifier using *send* primitive of the session manager. The application is now bound to the remote service (late binding) and session manager acts to start the session. For the purpose, a new session table entry is created and initialized, a new locally unique session identifier is assigned, and the submitted data is processed to be put into SeMA packets. Data is fragmented, and sequence numbers are associated if total data size exceeds the maximum possible payload size of one SeMA packet. Last packet (the only packet if no fragmentation took place) of the session always has its last flag set to indicate the number of fragments this session owns. Details of the structure of a SeMA packet are given in Section 4.3. This ready-to-go packets are placed into the sessions *departure buffer*. Routing agent of the host is responsible from making those packets their way to the destination. Mechanisms of routing are discussed in Section 4.4.3.

Session manager also informs the application of a successful delivery if termination packet of the remote party is received. Applications are to maintain their exception or timeout mechanisms to handle the absence of termination indication from session manager.

Since session manager keeps the information related to a given session in a table entry, together with all necessary buffers, it is possible to maintain multiple sessions (from different applications or from the same application). A session on SeMA network

may be uniquely identified by a <*localSessionId, hostAddr*> tuple. The same session is always uniquely addressed by its corresponding <*remoteSessionId, remoteAddr*> tuple.

4.4.2. Service Agent

The SeMA sessions are created, maintained, and terminated in a service-centric manner. To manage this, a separate functional entity, called service agent, is defined. Service agent maintains the service table, a vital data structure of the protocol, and performs service related message processing.

4.4.2.1. Service Table. Service table is the main data structure of many algorithms defined in the SeMA protocol stack. It is composed of entries that contain service instances available at remote hosts. The fields of the table and a sample entry is illustrated in Figure 4.8. The table is updated and new entries are added during the lifetime of the host. Service table is not only used by the service agent itself, but also used by the routing agent to extract valid source routes for a given service instance.

Service table is mainly populated by the service instances that are delivered by the *communication agent* as service announcement packets (See Section 4.3 for SeMA packets). Service announcements are broadcast messages forwarded through the network to have local service tables with up-to-date service information available on the network. Service agent may record the instance into the table if it satisfies some

SERVICE TABLE

Index	*Service Instance*	*Mobile Host Instance of Provider*	*Entry Arrvival*	*Entry Expiration*	*Hop Count*	*Mobile Host Instances of Forwarding Hosts*
84	<service name="printer"> <keyword attr="color">no</keyword> <keyword attr="postscript">yes</keyword> . . . </service>	<mobileHost WLID="A0:11:90:F2:43:D1"> <battery type="NiMH" max="100">34</battery> <security level="high"> <publicKey id="PGP">101 ... 00</publicKey> . . . </mobileHost>	*2003.02.14 13.30.59.01*	*2003.02.14 14.30.59.01*	*4*	. . . . .

Figure 4.8. Service table with a sample entry

conditions. These are discussed in the Section 4.4.2.2.

It is legal for service table to include more than one entry for a given service instance on a remote host. The alternative entries provide different multihop routes towards the provider of this specific instance. The host instances of the route from which this service instance carrying announcement packet is received is recorded into the table together with its hop count. It is also clear that this approach is consistent with the service-centric protocol in the sense that given two services provided by the same remote host is not necessarily reached via the same route (i.e., host or network feature aware routing).

Service table entries bear a timestamp that indicates the time that the entry is inserted. Also extracted from the service announcement packet, there is the entry expiration timestamp, determining the time that the service instance is valid as a working alternative.

Service table entries are accessed via their index numbers and this number is assigned at insertion automatically by the service agent. Index numbers are not reused and expired entries (according to their expiration information) are not removed as long as available space permits. Expired entries may be used for some optimization alternatives in case of route losses towards a specific host.

4.4.2.2. Handling Arriving Announcements. Upon receiving a SeMA packet of announcement type, the communication agent of the SeMA stack passes the packet to the service agent for processing. If it is the first time that this service instance is received from the provider host, a new service table entry is created, an index is assigned and the service instance is recorded with its provider and forwarder host instances. The arrival timestamp entry is filled with the current wall clock of the host, and the expiration timestamp is found by adding the validity interval (from the received packet) to this arrival timestamp. The reason for such a timestamp mechanism is explained in the Section 4.4.2.3.

If service agent finds one or more valid (not expired) entries for the same service instance from the same host, the announcement packet is processed in a different manner. If there is a *SEMARP_SERVICE_TABLE_ADDING_HOP_THRESHOLD* or less hop count carrying entry recorded, this new entry is treated as 'too expensive to record' in the sense that number of hops to reach the provider host is high compared to what is already in the table. This threshold value is a protocol parameter to be adjusted according to the node density of the target environment. If the arriving announcement hop count is suitable, it is recorded to the table and the announcement is passed to the routing agent for further relaying. Service agent does not relay the announcements that is not recorded because it is usually waste of bandwidth to further relay what is found to be 'worthless' to record. As a further emergency break, no more than *SEMARP_SERVICE_TABLE_SAME_SERVICE_LIMIT* number of entries pointing to the same service of the same host is recorded to the table, even if it is found feasible by the threshold value check. This feature is necessary to prevent an ever-growing service table in an environment where very densely deployed ad hoc network hosts exist (i.e., generating more than necessary route alternatives). If the received announcement is found as an entry in the table except its expiration timestamp, only a timestamp update is done by the service agent.

The arriving service announcement handling algorithm performed by the service agent is given in Figure 4.9.

4.4.2.3. Handling Departing Announcements. Applications are interfaced with *offer* primitive of the session manager for service starting purposes. The session manager asks service agent for announcement of this new service as explained in Section 4.4.1.1. Service agent first inserts this new service to its own service table, and prepares a SeMA packet of type *announcement*. The service XML instance provided by the application is put into the payload of the packet and this hosts XML instance is placed as the first host on the route, indicating this host as the provider of the service. The time interval value of the packet is filled with the difference of the absolute service validity time provided by the application and the current wall clock of the host. Therefore the packet carries

a relative time interval during which the service is provided by this host. This relative time interval is necessary to maintain a working distributed expiration mechanism without the assumption of centrally synchronized clocks of different mobile hosts. It is obvious that the ad hoc network would need a central clock synchronization method, if service agent inserts the absolute service lifetime timestamp into the announcement packet. By using a time interval in announcement packets, on the other hand, we ignore the time that the announcement packet spends during the flooding over the

```
handleArrivedServiceAnnouncement(SeMA_Packet announcement)

  SeMA_ServiceTableEntry stEntry;
  SeMA_HostInstance providerHost, hostOnRoute;
  SeMA_ServiceInstance announcedService;

    parseServiceXML(announcement.payload, &announcedService);
    parseHostXML(announcement.providerXML, &providerHost);

    If (providerHost.hostId == myInstance.hostId)
      \\ This is my service offering
      Set notToBeProcessed = TRUE; return;

    forall announcement.hostsOnRouteXML
      parseHostXML(announcement.hostsOnRouteXML, &hostOnRoute);
      If (hostOnRoute.hostId == myInstance.hostId)
        \\ Circular announcement flooding detected
        Set notToBeProcessed = TRUE; return;

    forall matching stEntry
      If (stEntry.hopCount + SEMARP_SERVICE_TABLE_ADDING_HOP_THRESHOLD
                                < announcement.hopCount)
        \\ It is not worth recording
        Set notToBeProcessed = TRUE; return;

    If numberof(matching stEntry) > SEMARP_SERVICE_TABLE_SAME_SERVICE_LIMIT
      \\ Have enough of this service already
      Set notToBeProcessed = TRUE; return;

    /* Yet another service instance is recorded to the table */
    recordIntoServiceTable(&announcement);
    /* ...  and it is further relayed to be flooded */
    sendRoutingAgent(&announcement);
    return;
```

Figure 4.9. The arriving service announcement handling algorithm

network. Upon reception, a service agent acts as if the service is valid for an interval starting at the moment of reception.

The algorithm that service agent uses to start offering services is given in Figure 4.10.

4.4.2.4. Fetching and Discovering Services. The fundamental purpose of the service agent is to find suitable service instances to the applications that used *fetch_service* primitive of the session manager. Service agent uses the service instance provided by the application as the basis of its decision.

First, the service agent performs a service name matching query on the service table among the entries that carry a valid (not expired) timestamp. For those entries that match the requested name, an attribute-value filtering is applied. Found entries are eliminated as long as they do not have the specified attribute-value pairs by the application. The attribute-value pairs specified by the application may be a subset of available services attribute-value pairs. These extra features are to be evaluated by the application itself. If at least one valid service instance is found satisfying the conditions above, its service table index is returned to the session manager, therefore to

```
handleDepartingServiceAnnouncement(SeMA_ServiceXML offeredService, timestamp validUntil)

  SeMA_Packet announcementPacket;

    Set announcementPacket.pktType = SeMA_ANNOUNCEMENT;
    Set announcementPacket.clientId = myInstance.hostId;
    Set announcementPacket.twOfServiceInst = validUntil - now();
    Set announcementPacket.payload = offeredService;
    generateHostXML(myInstance, &announcementPacket.hostsOnRoute[0]);

    /* Adding my own service to my own service table */
    recordIntoServiceTable(&announcementPacket);
    /* ...  let it start to be announced */
    sendRoutingAgent(&announcementPacket);
    return;
```

Figure 4.10. The departing service announcement handling algorithm

the application. If more than one matching instance is found, all of them are returned to the application for a further decision. Currently, this matching is done on per service instance of a host basis, meaning that only one entry is returned for a specific service instance on a specific host. This is because the populated entries for a given instance at a given host are there for routing purposes, and application, currently, do not select services on routing alternative basis. From the provided instances, application starts the ad hoc network by selecting one of them and binding itself by submitting its data to the session manager.

If no matching service instances are found on the local service table, the procedure called *service lookup* is initiated by the service agent. The purpose of the service lookup is to find a valid suitable service instance as quick as possible, preferably from near neighbors service tables. To initiate a service lookup, service agent prepares a SeMA packet of type *service lookup* and attaches the requested service instance received from the application. The service lookup packet is actually a broadcast message but is emitted to the network using an expanding ring approach to limit the unnecessary message flooding. The service lookup messages carry a time-to-live (TTL) value (literally in *sessionId* field of the SeMA packet) which is decremented at every forwarding hop and the message is no more relayed when TTL reaches zero. By adjusting the TTL value, the service looking host controls the diameter of the scope of the lookup request.

Communication agents of the hosts that receive a service lookup packet pass the packet to their service agents which then realize a similar check on their respective service tables. If at least one valid matching entry is found, the entry is sent back to the originator of the lookup request in a unicast *lookup reply* packet (i.e routing agent reverses the route that the respective lookup packet has arrived with). The answering service agent concatenates the route from this host to the service provider (using the service table entry) with the route on the lookup packet, so that the originator of the lookup will be able to find out how to reach the service provider host via this answering host. If no suitable entry is found locally and if TTL value permits, the lookup request is broadcast again to the network (with TTL value decremented by one). For the worst case, if such a service exists somewhere reachable in the network, the lookup request

reaches the provider of the service.

If the service lookup originator has not received a service lookup reply from some host on the network, its service agent re-broadcasts the message with an increased TTL value. Currently the lookup start diameter is determined by the *SEMARP_SERVICE_LOOKUP_EXPANDRING_START_DIAMETER* parameter and increased by *SEMARP_SERVICE_LOOKUP_EXPANDRING_STEPSIZE* amount of hops for every *SEMARP_SERVICE_LOOKUP_EXPANDRING_TIMEOUT* amount of time passed without a service lookup reply packet received. This ring is expanded for *SEMARP_SERVICE_LOOKUP_EXPANDRING_RETRIES* times at most before the application is informed of 'no such service available'.

Upon reception of a service lookup reply from a neighbor (near or far), the service agent behaves as if it has received a service announcement and places this new information to its service table and returns the index of this entry to the session manager in order to satisfy awaiting application fetch request.

For our proposed protocol, there is a trade off between number of service announcement packets and number of service lookup or lookup reply packets. If scope of announcement flooding is limited, then service lookup actions are expected to be frequent. Currently flooding of service announcements are not limited, therefore resulting in less lookup need and low latency in service discovery.

The summary of the operations to find a service is given as the *findService* algorithm in Figure 4.11.

4.4.3. Routing Agent

Routing agent is the component of the protocol that tries to attach valid routes to new SeMA packets, forward the ones attached with routes, and heal the ones that have lost their way to destination because of link breaks. All entities of the protocol ask routing agent to have a SeMA packet sent out of the host.

4.4.3.1. Routing Strategy. For unicast packets of the SeMA protocol, source routing is used to send packets towards their destinations. Different from other source routing approaches, like Dynamic Source Routing (DSR) [23], source route discovery is not defined as a separate process but already realized with the service announcement mechanism. Routing agent is provided with SeMA packets from session manager that are already bound to a service instance available in the service table of the host. Routing

```
findService(SeMA_ServiceInstance askedService)

  SeMA_Packet lookupPacket;
  SeMA_ServiceTableEntry stEntry;

    forall stEntry
      If ((stEntry.serviceName == askedService.serviceName) & &
          (stEntry.attribute/values ⊇ askedService.attribute/values))
        Set numberOfSuitableInstances++;
        addToFoundList(stEntry);

    If (numberOfSuitableInstances > 0)
      sendSessionManager(FoundList); return;
    Else
      /* ...  start service lookup */
      Set numberOfLookupAttempts++;
      Set lookupPacket.pktType = SeMA_LOOKUP;
      Set lookupPacket.clientId = myInstance.hostId;
      Set lookupPacket.sessionId = SEMARP_SERVICE_LOOKUP_EXPANDRING_START_DIAMETER;
      Set lookupPacket.payload = askedService;
      sendRoutingAgent(&lookupPacket);

    Wait Until
      lookupReply received:
        recordIntoServiceTable(&lookupReply);
        sendSessionManager(newServiceIndex); return;
      timeout in SEMARP_SERVICE_LOOKUP_EXPANDRING_TIMEOUT received:
        If (numberOfLookupAttempts < SEMARP_SERVICE_LOOKUP_EXPANDRING_RETRIES)
          /* increment expanding ring diameter and repeat lookup */
          Set lookupPacket.sessionId += SEMARP_SERVICE_LOOKUP_EXPANDRING_STEPSIZE;
          sendRoutingAgent(&lookupPacket);
          goto(WaitUntil);
        Else
          sendSessionManager(timeoutIndication); return;
```

Figure 4.11. The steps performed to find a suitable service by the service agent

agent simply extracts the route (i.e., the sequence of hosts that forwarded this service instance at the time of announcement flooding) from the corresponding service table entry and inserts it into the *hostOnRoute* field of the outgoing SeMA packet.

This simplifying approach to routing has its disadvantages on the other hand. The route information found at the local service table may be out-of-date because of the highly varying node mobility at the target environment. Routing agent is equipped with some basic healing algorithms that try to heal a route on a SeMA packet that is not valid anymore. These are given in Section 4.4.3.2. The service announcement frequency on the ad hoc network has an important effect on routing performance in this sense.

For the sake of bandwidth efficient routing process, routing agent maintains a data structure called *session cache*. Routing agent records identifier parts of all packets it has processed to this session cache. Session cache is maintained in FIFO manner and its size may be determined according to the memory space available. Routing agent uses the session cache to immediately discard the packets that it has previously processed. Furthermore, if the routing agent forwards a terminate packet for a given session between two hosts, it invalidates the entries (i.e., mark them) corresponding to this session to avoid further processing of any zombie packets (belonging to this terminated session) travelling the network. Space dedicated as the session cache is to be determined at a size that is small enough to catch the double processing of some flooded data packets. Having larger sized session cache will do no harm in that sense but will slow down the session cache search procedure. The session cache with sample entries is illustrated in Figure 4.12.

As the forwarder of packets to their destinations, routing agent is the place where communication agent delivers all *data* packets it receives. If routing agent has not seen the new arrived data packet, it is first placed into the session cache, and discarded otherwise. If the packet is not destined for this host, routing agent locates its own instance from the source route and extracts the next hop id from the available source route. Packet is then forwarded to this next hop via the communication agent. If this

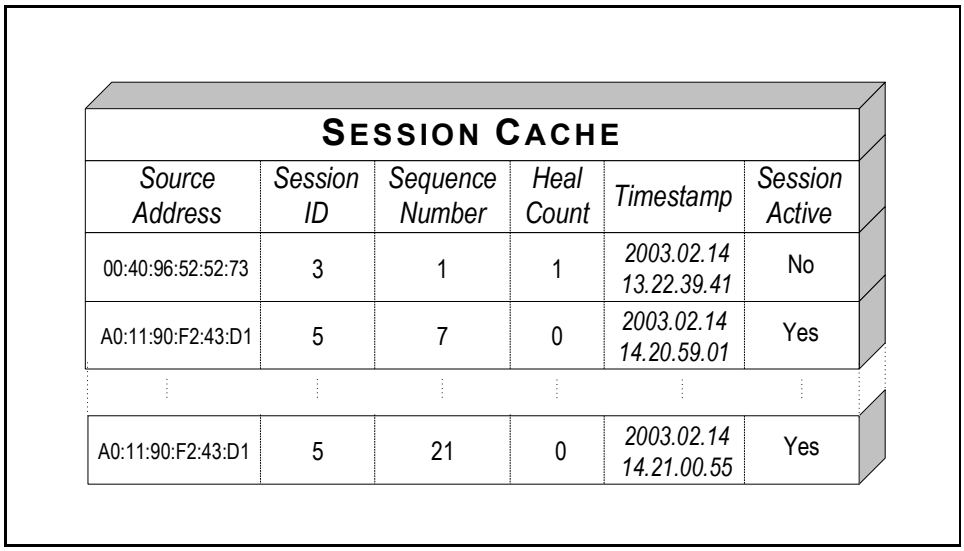


Figure 4.12. Session cache with sample entries

host is the intended recipient of the packet, the packet is passed to the session manager for further processing. Exceptions of this process is the unavailability of the next hop specified in the source route of the packet. The route loss and healing is explained in the next section.

4.4.3.2. Route Loss and Healing. Whenever a unicast packet is given to the underlying data link layer for transmission by the communication agent, it is probable that the next hop will not be in the wireless transmission range of the host transmitting the packet. Additionally, the next hop may be in the transmission range of the transmitting host, but transmission may be unsuccessful because of various reasons, such as fading, interference related bit errors etc. For those cases, the data link layer returns a link failure indication (as assumed in Section 4.1), so that SeMA can act accordingly.

If a link failure indication is received for a SeMA packet, routing agent performs a local search on its service table to extract an alternative valid minimum hop source route towards the destination of the failed packet. Next hop available in this new route, obviously, needs to be different from the one that has been previously tried. Routing agent retries this route healing *SEMARP_LOST_ROUTE_HEAL_TRY_LIMIT* times and keep track of status of a packet from its respective session cache entry (*Heal Count* field in Figure 4.12). This behavior of SeMA routing is again different from DSR, where

source of the respective lost packet is informed of route loss by means of *route error* messages (route salvaging is a similar optimization defined in DSR, but source host is notified of loss anyhow).

If packet has not been successfully forwarded to the next hop along the source route after *SEMARP_LOST_ROUTE_HEAL_TRY_LIMIT* times healing, or no suitable route alternatives are found from the local service table, routing agent switches to broadcasting the packet by setting its *lost flag*. Any routing agent that receives this lost packet processes it even if its not among the hosts on the source route. If the routing agent that received the lost packet finds itself on the source route, the lost flag of packet is reset and routing switches to the normal procedure. Otherwise the same healing procedure is applied for the packet. Further broadcast of the packet on failure is a protocol parameter and currently flooding is not continued after the first broadcast. The probability of route healing is higher if broadcast is continued but it will cause more contention on the network, especially if the network is heavily loaded. Routing and healing algorithm for a given SeMA packet is summarized in Figure 4.13.

Route loss actions and corresponding healing procedures are only taken for *data* and *terminate* packets. Other packets are broadcast type packets (*announcement* and *lookup*) or *lookup reply* packets and their loss is not vital in the sense that alternatives of the packets are expected to arrive from different routes anyway.

Routing agent is the place where different alternative routing approaches are possible to implement. For example, specific to a given service class, routing agent may try to select the route in a different manner from its inventory (i.e., service table entries) such as the minimum hop including or maximum battery capacity holding etc.

4.4.4. Communication Agent

Communication agent functions as a simple wrapper to the primitives provided by underlying data link layer. Data link primitives are abstracted away from the entities

of SeMA by the communication agent.

Communication agent dispatches the arriving data link frames as SeMA packets and delivers them accordingly. *Announcement*, *lookup* and *lookup reply* packets are

```
routeSeMAPacket(SeMA_Packet packetToBeProcessed)

  SeMA_ServiceTableEntry stEntry;
  SeMA_HostInstance hostOnRoute;

  If (!packetHasSeenBefore(&packetToBeProcessed))
    /* First, record the packet to seen list */
    insertSessionCache(&packetToBeProcesed.header);

    If (!packetToBeProcessed.lostFlag)
      /* A regular SeMA packet to be routed */
      forall packetToBeProcessed.hostsOnRouteXML[i]
        parseHostXML(packetToBeProcessed.hostsOnRouteXML[i], &hostOnRoute);
        If (hostOnRoute.hostId == myInstance.hostId)
          /* Here is my address, let us extract the next hop */
          parseHostXML(packetToBeProcessed.hostsOnRouteXML[i+1], &hostOnRoute);
          Set nextHopId = hostOnRoute.hostId;
          sendCommunicationAgent(&packetToBeProcessed, nextHopId);
          return;
    Else
      /* This packet has lost its way towards destination */
      If (!packetHasBeenHealedEnough(&packetToBeProcesed.header))
        /* Let us look an alternative route entry for this lost packet */
        forall stEntry
        If (stEntry.provider.hostId == packetToBeProcesed.destinationHostId)
          updateSourceRoute(&packetToBeProcesed, stEntry.index);
          Set packetHealed = TRUE;
          sendCommunicationAgent(&packetToBeProcessed, newNextHopId);
          return;
      Else
        /* Packet has been healed SEMARP_LOST_ROUTE_HEAL_TRY_LIMIT times, switch to flooding */
        Set packetToBeProcessed.lostFlag = TRUE;
        sendCommunicationAgent(&packetToBeProcessed, BROADCAST);
        return;
  Else
     This packet has been seen before */
    Set packetToBeDiscarded = TRUE;
    return;
```

Figure 4.13. The steps performed to route a SeMA packet

delivered to service agent, *data* and *terminate* packets are delivered to the routing agent of the host by the communication agent.

Although SeMA packets are currently not fragmented to be carried in more than one data link layer frame, communication agent is the place to realize this feature if it is desired in the future.

4.5. Illustrative Examples

In this section, some illustrative examples are presented to clarify basics of the algorithms given in the preceding sections. Examples are given on a sample ad hoc network topology, composed of nodes running proposed protocol stack.

In Figure 4.14, service announcement procedure is illustrated. There are four nodes in the example with their wireless transmission ranges indicated with circles bearing their host identifiers. For this example, Node 1 provides a printer service for the network and starts to announce this service as illustrated in Part (a) of Figure 4.14. The announcement packet is only received by the Node 2 since it is the only node within the transmission range of Node 1. Upon receiving the announcement packet from Node 1, Node 2 records availability of the new service and starts to relay the announcement by broadcasting the announcement. Before relaying the announcement, Node 2 inserts its own host instance to the source route of the packet. The announcement of Node 2 is received by Node 1, Node 3 and Node 4 as illustrated in the Part (b) of Figure 4.14. Node 1 immediately discards the received announcement since it carries the service provided by itself. Node 3 and Node 4 record this announcement as a service that is reachable via Node 2 and offered by Node 1. As Node 3 and Node 4 further relay the received announcement (Part (c) of Figure 4.14), Node 2 receives two announcements that should be discarded. This is because Node 2 already appears on the source route field of the announcements received from Node 3 and Node 4. However, Node 3 records the announcement received from Node 4 and Node 4 records the announcement received from Node 3. This creates two alternative entries at Node 3 and Node 4 regarding the same service instance of Node 1. If the

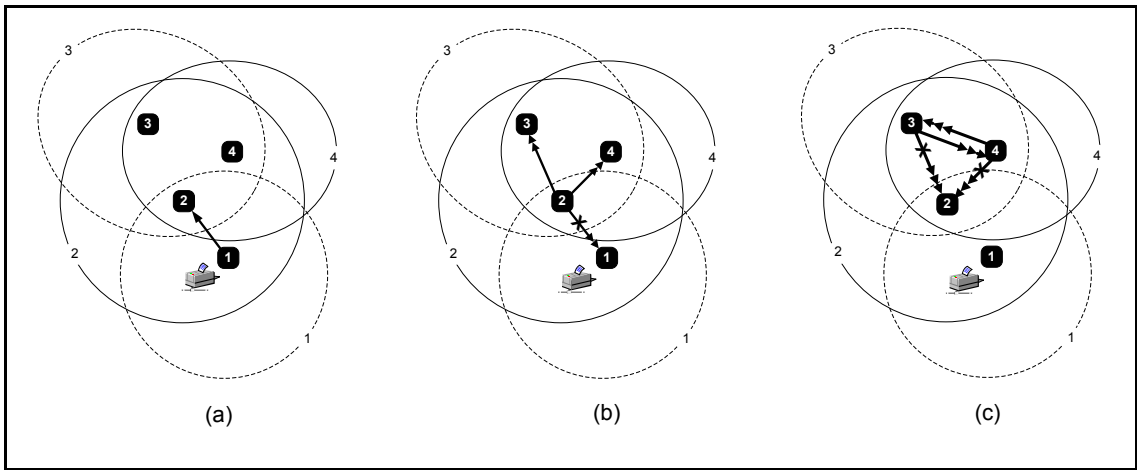
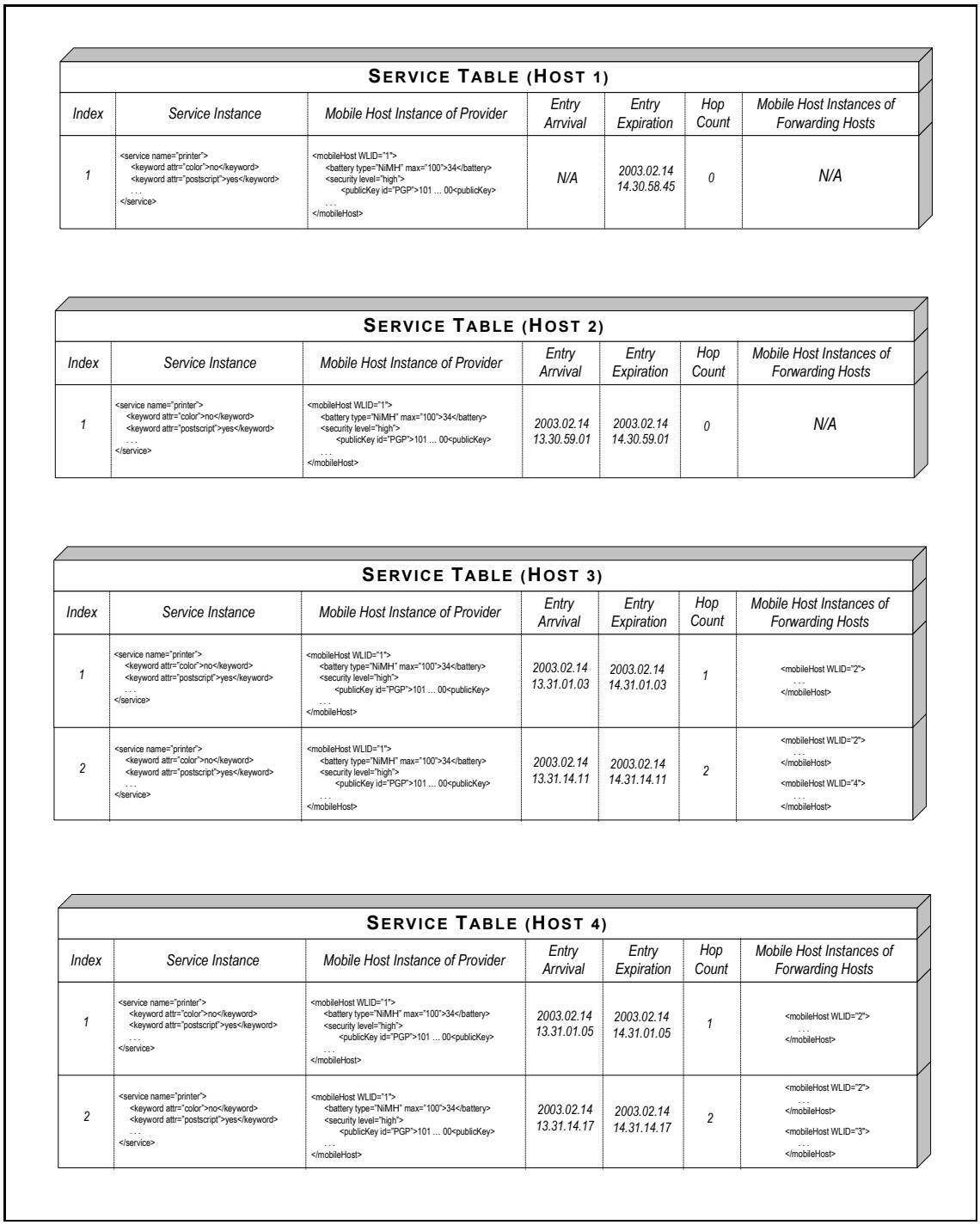


Figure 4.14. Service announcement procedure in a sample network

SEMARP_SERVICE_TABLE_ADDING_HOP_THRESHOLD parameter of the protocol is set to zero, Node 3 would not record the announcement received from Node 4 since there is already an entry with smaller hop count. After these steps of the service announcement, service tables of the nodes are found as illustrated in Figure 4.15. Details of service announcement procedure are given in Section 4.4.2.

Just after the announcement procedure is over, two more nodes (Node 5 and Node 6) join to the sample network topology as illustrated in Part (a) of Figure 4.16. As soon as Node 6 needs a printer service, a lookup procedure has to be started, since Node 6 has not yet heard of an announcement. The first lookup message is broadcast with expanding ring diameter being equal to one. Only Node 5 receives this lookup request as seen in Part (b) of Figure 4.16. Node 5 has no suitable instance to reply to this request and can not further relay the message (because of the TTL value). After *SEMARP_SERVICE_LOOKUP_EXPANDRING_TIMEOUT* amount of time without a positive reply, Node 6 re-initiates the lookup for the second time with the expanding ring diameter being equal to two. This time, lookup request packet reaches to Node 3 which then replies with a lookup reply packet including the service instance recorded in its service table with index number (1). Node 5 takes its role in relaying the unicast lookup reply packet to Node 6. Finally, service table of the Node 6 is populated with an entry as illustrated in Figure 4.17.

After the announcement and lookup phases, an application on Node 3 initiates a service fetch for a printer and immediately finds the printer on Node 1 (from its local service table). The returned service table index for this application is (1), since it is the minimum hop path via Node 2. The application starts the session and SeMA packets are routed through Node 2 towards the destination Node 1 as illustrated in Part (a)



SERVICE TABLE (HOST 1)

Index	Service Instance	Mobile Host Instance of Provider	Entry Arrvival	Entry Expiration	Hop Count	Mobile Host Instances of Forwarding Hosts
1	<service name="printer"> <keyword attr="color">no</keyword> <keyword attr="postscript">yes</keyword> . . . </service>	<mobileHost WLID="1"> <battery type="NiMH" max="100">34</battery> <security level="high"> <publicKey id="PGP">101 ... 00<publicKey> . . . </mobileHost>	N/A	2003.02.14 14.30.58.45	0	N/A

SERVICE TABLE (HOST 2)

Index	Service Instance	Mobile Host Instance of Provider	Entry Arrvival	Entry Expiration	Hop Count	Mobile Host Instances of Forwarding Hosts
1	<service name="printer"> <keyword attr="color">no</keyword> <keyword attr="postscript">yes</keyword> . . . </service>	<mobileHost WLID="1"> <battery type="NiMH" max="100">34</battery> <security level="high"> <publicKey id="PGP">101 ... 00<publicKey> . . . </mobileHost>	2003.02.14 13.30.59.01	2003.02.14 14.30.59.01	0	N/A

SERVICE TABLE (HOST 3)

Index	Service Instance	Mobile Host Instance of Provider	Entry Arrvival	Entry Expiration	Hop Count	Mobile Host Instances of Forwarding Hosts
1	<service name="printer"> <keyword attr="color">no</keyword> <keyword attr="postscript">yes</keyword> . . . </service>	<mobileHost WLID="1"> <battery type="NiMH" max="100">34</battery> <security level="high"> <publicKey id="PGP">101 ... 00<publicKey> . . . </mobileHost>	2003.02.14 13.31.01.03	2003.02.14 14.31.01.03	1	<mobileHost WLID="2"> . . . </mobileHost>
2	<service name="printer"> <keyword attr="color">no</keyword> <keyword attr="postscript">yes</keyword> . . . </service>	<mobileHost WLID="1"> <battery type="NiMH" max="100">34</battery> <security level="high"> <publicKey id="PGP">101 ... 00<publicKey> . . . </mobileHost>	2003.02.14 13.31.14.11	2003.02.14 14.31.14.11	2	<mobileHost WLID="2"> . . . </mobileHost> <mobileHost WLID="4"> . . . </mobileHost>

SERVICE TABLE (HOST 4)

Index	Service Instance	Mobile Host Instance of Provider	Entry Arrvival	Entry Expiration	Hop Count	Mobile Host Instances of Forwarding Hosts
1	<service name="printer"> <keyword attr="color">no</keyword> <keyword attr="postscript">yes</keyword> . . . </service>	<mobileHost WLID="1"> <battery type="NiMH" max="100">34</battery> <security level="high"> <publicKey id="PGP">101 ... 00<publicKey> . . . </mobileHost>	2003.02.14 13.31.01.05	2003.02.14 14.31.01.05	1	<mobileHost WLID="2"> . . . </mobileHost>
2	<service name="printer"> <keyword attr="color">no</keyword> <keyword attr="postscript">yes</keyword> . . . </service>	<mobileHost WLID="1"> <battery type="NiMH" max="100">34</battery> <security level="high"> <publicKey id="PGP">101 ... 00<publicKey> . . . </mobileHost>	2003.02.14 13.31.14.17	2003.02.14 14.31.14.17	2	<mobileHost WLID="2"> . . . </mobileHost> <mobileHost WLID="3"> . . . </mobileHost>

Figure 4.15. Service tables of hosts after the announcement

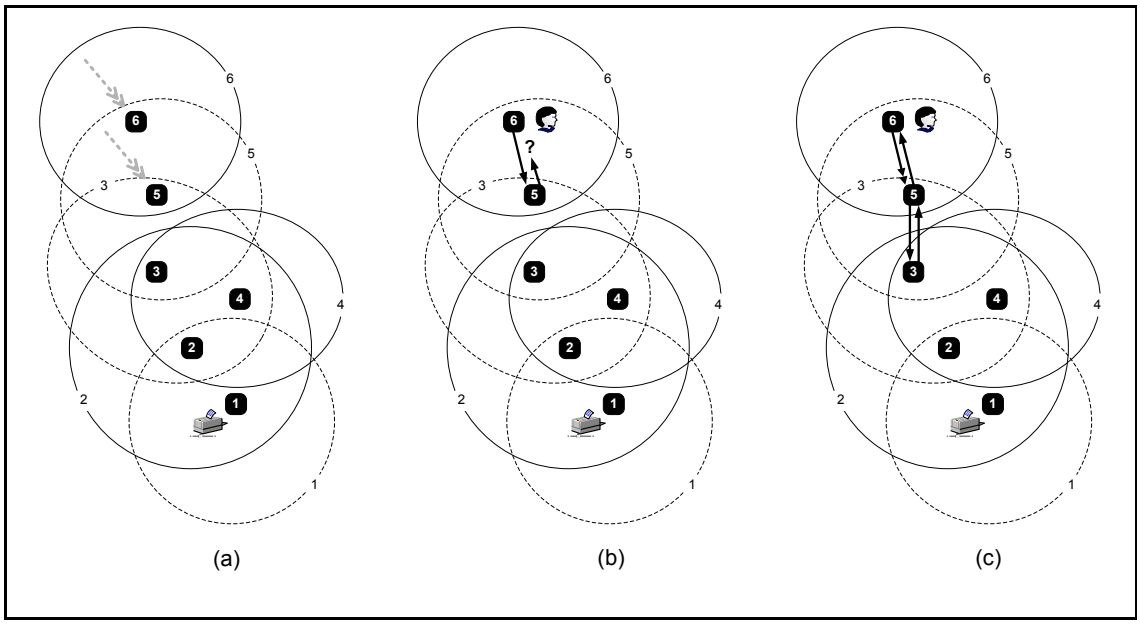


Figure 4.16. Service lookup procedure in a sample network

of the Figure 4.18. During the session, Node 3 starts to move towards the direction indicated by the gray arrow in Figure 4.18 Part (a). When Node 3 arrives in its new place, as shown in Part (b) of Figure 4.18, Node 2 is not reachable anymore. Therefore Node 3 initiates the route healing process (after the link layer failure indication) explained in Section 4.4.3.2. Finding the alternative entry for the same service from the service table (See Figure 4.15 for the tables), Node 3 switches the route to include Node 4 and then Node 2. While the new route is in effect, unfortunately, Node 2 goes out of battery and stops relaying packets towards Node 1. Having no other route alternatives, Node 4 switches to flooding the remaining data packets. Luckily, Node 1 is now in the transmission range of Node 4 and it receives the flooded data packets.

SERVICE TABLE (HOST 6)

Index	*Service Instance*	*Mobile Host Instance of Provider*	*Entry Arrvival*	*Entry Expiration*	*Hop Count*	*Mobile Host Instances of Forwarding Hosts*
1	<service name="printer"> <keyword attr="color">no</keyword> <keyword attr="postscript">yes</keyword> . . . </service>	<mobileHost WLID="1"> <battery type="NiMH" max="100">34</battery> <security level="high"> <publicKey id="PGP">101 ... 00</publicKey> . . . </mobileHost>	*2003.02.14 13.35.43.02*	*2003.02.14 14.30.02.01*	*3*	<mobileHost WLID="2"> . . . </mobileHost> <mobileHost WLID="3"> . . . </mobileHost> <mobileHost WLID="5"> . . . </mobileHost>

Figure 4.17. Service table of Node 6 after lookup reply procedure

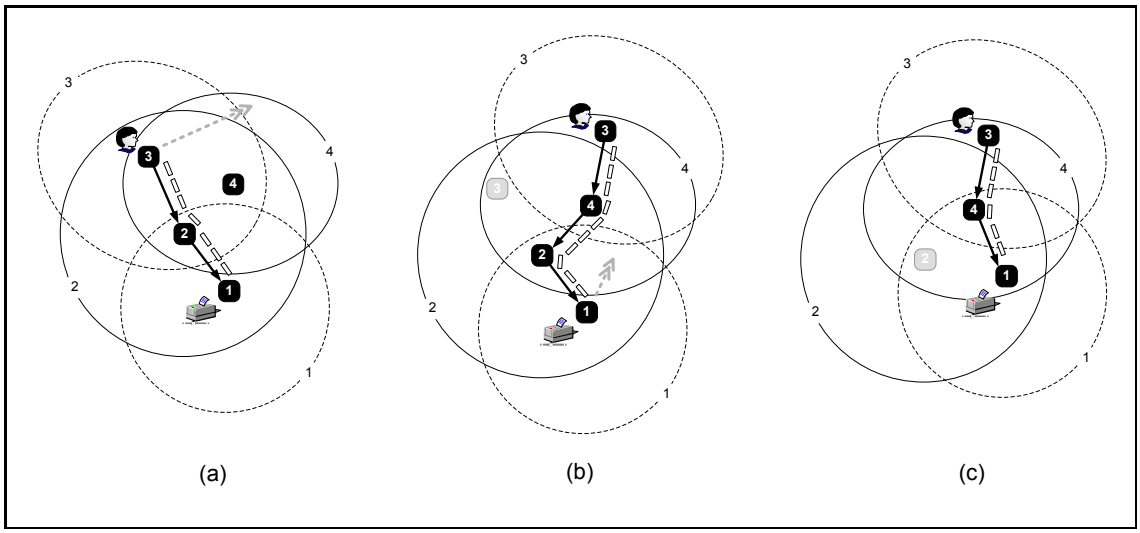


Figure 4.18. SeMA packet routing procedure in a sample network

Therefore, session continues and finishes intact. Details of routing and route healing are given in Section 4.4.3.

5. IMPLEMENTATION OF SIMULATION

Simulation study of the proposed protocol is conducted using GloMoSim (Global Mobile Information Systems Simulation Library), a simulation library developed at UCLA Parallel Computing Laboratory [68]. GloMoSim has been built to provide a scalable simulation environment with support for wireless networks (wired network support was planned as a future extension) using a layered approach similar to 'OSI seven layer network architecture'. Standard APIs have been used between those layers using parallel discrete event simulation capability provided by Parsec [69], which is a general purpose C-based simulation language for sequential and parallel execution of discrete event simulation models. An enhanced commercial version of the GloMoSim is developed and marketed by Scalable Network Technologies Inc. as Qualnet [70].

Possible simulator software alternatives that could have been used in simulations of the proposed protocol were OPNET Modeler [71] from OPNET Technologies Inc., and ns-2 [72] (developed in collaboration with many institutions and researchers). OPNET was not preferred because of licensing issues (i.e., it was not available free of charge for research institutions). Although ns-2 is being widely used for wireless network simulations in the research communities, it has not been preferred because of the following reasons. The effort involved in learning and using the simulator is comparably high, and ns-2 is more useful if lower layer protocol statistics are of interest. Ns-2 has originally been developed to be a wired network simulator and later extended to include wireless networking support. Additionally, feasibility of using ns-2 diminishes as the scale of the network under simulation is increased (in terms of the size of the area, the number of nodes, and the amount of application traffic generated).

Following sections provide a brief introduction to the details of the design of the GloMoSim and explain the newly added components to the simulator.

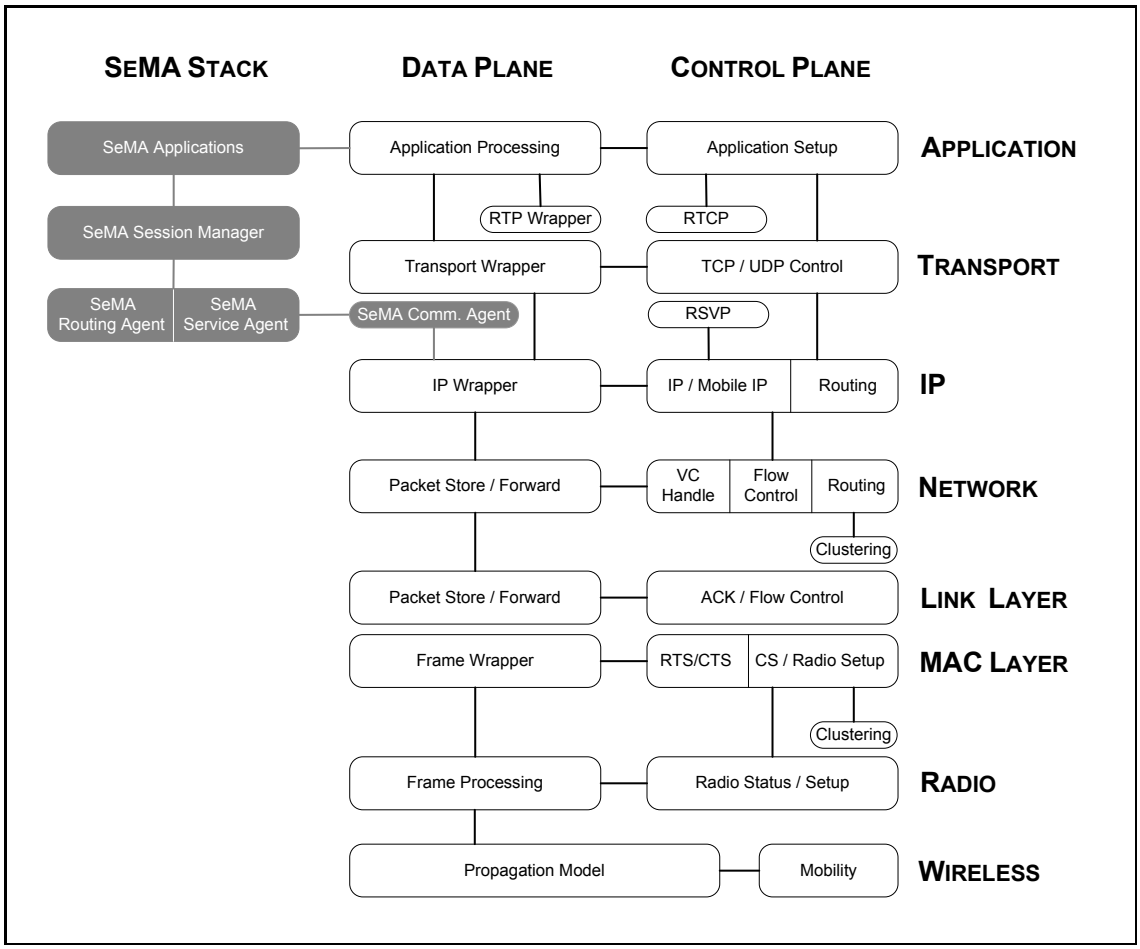


Figure 5.1. GloMoSim layers and implemented SeMA stack

5.1. Structure of the GloMoSim

GloMoSim is composed of a set of layers, each with its own API. Models of protocols at one layer interact with those at a lower (or higher) layer only via these APIs. This allows easy integration of new models into the simulator by different people. Decomposition of those layers is illustrated in Figure 5.1. The addition of proposed protocol to the original structure is emphasized with gray shaded boxes in the figure. An example to API calls is given in Figure 5.2. Given example calls are used to handle data packets between network and MAC layers. Using the first call in Figure 5.2, IP protocol requests a network layer datagram to be delivered in a MAC layer frame. The second call is made by the MAC layer to inform the network layer about a new datagram arrival.

```
void NetworkIpSendPacketToMacLayer(GlomoNode* node, Message* msg,
                                   InterfaceIdType interfaceId, NODE_ADDR nextHop)

void NetworkIpReceivePacketFromMacLayer(GlomoNode* node, Message* msg,
                                        NODE_ADDR lastHopAddress)
```

Figure 5.2. An example to API calls between layers in GloMoSim

GloMoSim simulation engine is constructed using Parsec, which has a message based approach to discrete-event simulation. In Parsec, physical processes are modelled by simulation objects called *entities*. GloMoSim uses Parsec entities to be the partitions of the terrain to be simulated. Although current distribution of GloMoSim (version 2.02) does not support partitioning, the parallel execution feature provided by Parsec may be used to distribute simulation of terrain partitions to different computers in the future. This would allow GloMoSim to scale to a considerably high number of nodes in simulations. Events of the simulation are represented by transmission of time-stamped messages within or among corresponding entities.

An initialization entity, called *driver*, exists in GloMoSim and is functionally similar to the *main* function of a program written in C. Parsec calls and entities are converted to valid C codes by the Parsec compiler (*pcc*), before all the simulation code is compiled with a generic C compiler (e.g., *gcc* for Linux). In GloMoSim, parsec source files have filename extensions of *.pc* before they are compiled.

There are many wireless protocols and models supported by GloMoSim. Many of them are implemented by GloMoSim development team and the rest comes from various sources (mainly from the developers of that specific protocol). Those currently available protocols and models for GloMoSim are listed below:

- **Application**: Telnet and FTP (both using TCPLIB), CBR (Constant Bit Rate), HTTP (using pre-collected behaviors), Web Phone, Web Caching[1]
- **Transport**: TCP (FreeBSD), NS TCP (Tahoe), UDP, DBS satellite models
- **Multicast Routing**: ODMRP, CAMP[1], AMRIS[1], AMRoute[1], AST[1], DVMRP[1]
- **Unicast Routing**: Distributed Bellman-Ford, Flooding, Fisheye, AODV, DSR,

Table 5.1. GloMosim directory structure

Directory Name (relative)	**Contents**
/bin	simulation executable, input and output files
/doc	includes documentation
/include	common include files
/main	contains the basic framework design
/scenarios	contains various sample configuration topologies
/java_gui	contains code for java visualization tool
/tcplib	contains TCP application communication patterns
/radio	contains code for the physical layer
/mac	contains code for the MAC layer
/network	contains code for the network layer
/transport	contains code for the transport layer
/application	contains code for the application layer

DSDV, OSPF, WRP, LAR, NS-DSDV, DREAM[1], MMWN[1]

- **MAC**: CSMA, FAMA, MACA, IEEE 802.11, MACA-W[1]
- **Radio**: DSSS Radio with or without capturing feature
- **Propagation**: Free Space, Rayleigh, Ricean, 2-Ray, SIRCIM, External Path-Loss Trace Files
- **Mobility**: Random Drunken, Random Waypoint, ECRV[1], Group Mobility[1], External Mobility Trace Files

5.1.1. Organization of Directories

GloMoSim distribution reflects the layered structure of the simulator. Table 5.1 gives the directories and their content classifications as they are found in the distribution. In the table and for the rest of the text, directory and filenames are given relative to the GloMoSim installation path, and the name of the main installation directory is always omitted.

[1]Not implemented by the simulator development team

```
                                                    ...from the file include/api.h
struct glomo_node_str {
  ...
  NODE_ADDR    nodeAddr; /* the network address of the node */
  ...
  int numberRadios;
  int numberInterfaces;
  GlomoProp       *propData; /* propagation information */
  GlomoRadio*     radioData[MAX_NUM_RADIOS]; /* radio layer information */
  GlomoMac*       macData[MAX_NUM_INTERFACES]; /* mac layer information */
  GlomoNetwork    networkData; /* network layer information */
  GlomoTransport  transportData; /* transport layer information */
  GlomoApp        appData; /* application layer information */
  ...
}GlomoNode;
```

Figure 5.3. An excerpt from GlomoNode structure definition

5.1.2. Representation of Layers

An important data structure in GloMoSim is *GlomoNode*, which represents hosts in the network. Rest of the protocol and model related structures (other than some global structures for simulation maintenance) are embedded in GlomoNode structure. This structure is organized to include each node's own stack of protocols (its parameters and statistics). The *nodeAddr* field of the structure constitutes the unique node identifier, used throughout the simulation code to refer to a specific node instance. An excerpt from the structure is given in Figure 5.3. Each layer in the GlomoNode structure has its own data structure (e.g., GlomoMac or GlomoNetwork) to represent corresponding protocols. To provide a further insight, the GlomoMac structure in the GlomoNode definition is expanded in Figure 5.4. Such layer structures usually home pointers to the structures of specific protocols, layer statistics and details of layer mechanisms.

5.1.3. Messages and Events

In GloMoSim, messages travel between layers of a given node or between layers of different nodes. In terms of their functions, following categorization is true for messages

```
                                        ...from the file include/mac.h
  struct glomo_mac_str {
    MAC_PROTOCOL macProtocol;
    int  interfaceIndex;
    int  bandwidth;
    int  radioNumber;
    BOOL   macStats;
    BOOL   promiscuousMode;
    clocktype  propDelay;
    void   *macVar;
}GlomoMac;
```

Figure 5.4. An excerpt from GlomoMac structure definition

in GloMoSim:

- **Non-Packet Messages**: Messages that are used to indicate events. They are used between layers (e.g., channel sensed busy message from radio to MAC layer) and within layers (e.g., timers)
- **Packet Messages**: Actual packets (e.g., IP datagrams) that eventually will appear on the wireless medium.

In order to create a message, its parameters should be set accordingly. Parameters of a message are listed as:

- **Destination**: Destination node ID, target layer in the destination node, protocol at that layer (if applicable), interface for that protocol (if applicable)
- **Event Type**: A unique event identifier
- **Message Info**: A customizable area to carry requested information
- **Payload**: Actual data that the message has to carry
- **Current Header Position**: A pointer to the current header of the message

To clarify the use of messages in GloMoSim, a non-packet message example (a timer event within MAC layer) and a packet message example (a MAC layer frame to radio layer) will be given (assuming IEEE 802.11 protocol is the objective). Although examples are selected from 802.11 protocol of MAC layer, the same methods of message

```
...
/* First, a message variable is declared */
Message *newMsg;

/* Then, its owner node, destination layer, destination protocol and event type is set */
newMsg = GLOMO_MsgAlloc(node, GLOMO_MAC_LAYER, MAC_PROTOCOL_802_11, MSG_MAC_TimerExpired);

/* Here, MAC interface of the message is set */
GLOMO_MsgSetInstanceId(newMsg, M802->myGlomoMac->interfaceIndex);

/* A custom size info space is reserved into the message */
GLOMO_MsgInfoAlloc(node, newMsg, sizeof(M802->timerSequenceNumber));

/* Then the reserved info space is filled with the desired custom information */
*((int*)(newMsg->info)) = M802->timerSequenceNumber;

/* Finally the message is scheduled to be sent to the destination node
   after a timerDelay amount of time */
GLOMO_MsgSend(node, newMsg, timerDelay);
...
```

Figure 5.5. Scheduling a self-timer event within the MAC layer

processing are applicable for the layers above or below.

Scheduling a timer event within the MAC Layer is illustrated in Figure 5.5. A message variable of type *Message* is declared and its parameters are set using *GLOMO_MsgAlloc* and *GLOMO_MsgSetInstanceId* calls. Then a special information that will be used when the timer expires is embedded into the *info* field of the message. Finally the timer message is scheduled for delivery by using *GLOMO_MsgSend* call. The *timerDelay* parameter specified in the call determines the time that the timer will go off, meaning that the destination layer (same layer for this example) is honored with the message. For this simple timer message example, payload portion of the message is not allocated and used. In order to handle the event scheduled in Figure 5.5, MAC layer event handling code should be expecting an event of type *MSG_MAC_TimerExpired*. Event handling routines for each layer are registered to GloMoSim and whenever there is a message to be processed *GLOMO_CallLayer* function extracts the target layer from the message. Then, the target layer event handling routine is triggered. A code snippet from the 802.11 protocol event handling routine is given in Figure 5.6 to show how this

```
void Mac802_11Layer(GlomoNode *node, int interfaceIndex, Message *msg) {
  ...
  switch (msg->eventType) {
    ...
    case MSG_MAC_TimerExpired:
      Mac802_11HandleTimeout(node, M802);
    case MSG_MAC_TimerExpired_PCF:
      Mac802_11HandleTimeout_PCF(node, M802);
    ...
  }
  GLOMO_MsgFree(node, msg);
  ...
}
```

Figure 5.6. Handling a timer event within the MAC layer

processing is done.

Similarly, preparing a MAC frame (802.11 frame for the example) out of the node is illustrated in Figure 5.7. A message is allocated and expanded to be a packet by *GLOMO_MsgPacketAlloc* call. After the MAC protocol header is added to the packet, it is sent out of the layer destined to the radio layer via *StartTransmittingPacket* call. Handling of packet messages are realized in the corresponding layers packet handling functions. Upon reception from a packet from radio layer, for example, 802.11 protocol packet handling routine *Mac802_11ReceivePacketFromRadio* is triggered for the further handling of the arriving packet. An illustrative excerpt from this routine is given in Figure 5.8.

Although it is possible to send any message to any node and any layer therein, packet messages must follow the usual track through the layered design of the simulation architecture. For example, under normal circumstances, network layer have to interact with the API provided by the MAC layer to send out a datagram encapsulated in a MAC frame. Therefore, the network layer must not use the calls that prepare a MAC frame and send the message directly to the radio layer.

```
...
/* First, a message to hold the packet is allocated */
Message *pktToRadio = GLOMO_MsgAlloc(node, 0, 0, 0);

/* Then, packet header information is prepared */
hdr.frameType = M802_11_DATA;
hdr.sourceAddr = node->nodeAddr;
hdr.destAddr = destAddr;
...

/* Here, a MAC frame is allocated into the packet and filled with information */
GLOMO_MsgPacketAlloc(node, pktToRadio, topPacket->packetSize);
memcpy(pktToRadio->packet, topPacket->packet, topPacket->packetSize);

/* MAC frame header is allocated and prepared header is placed into */
GLOMO_MsgAddHeader(node, pktToRadio, sizeof(M802_11FrameHdr));
memcpy(pktToRadio->packet, &hdr, sizeof(M802_11FrameHdr));

/* Then the packet is sent for transmission */
StartTransmittingPacket(node, M802, pktToRadio, M802_11_DIFS);
...
```

Figure 5.7. Preparing and sending a MAC frame to radio layer

5.1.4. Operation of Network Layer

Understanding GloMoSim network layer and its internal design is vital for two important reasons. First, many of the protocols available for ad hoc networks deal with the routing of packets in the network. GloMoSim network layer is the place where those routing protocols are invoked to function. Second and the more important reason is related to our protocol proposal. The simulation code of the protocol components is attached to the network layer and does not use any of the existing protocols above the network layer.

Currently, GloMoSim has only one option as a network layer protocol, which is the well-known Internet Protocol (IP). In the simulation, the proposed ad hoc networking protocol receives the packet delivery service from the network layer, not directly from a data link and MAC layer protocol (i.e., IEEE 802.11). The basic motivation behind this is to keep simulation code independent from the underlying MAC layer protocol

```
void Mac802_11ReceivePacketFromRadio(GlomoNode *node, Message *msg) {
  ...
  if (hdr->destAddr == node->nodeAddr)
    switch (hdr->frameType) {
      case M802_11_RTS:
        ...
      case M802_11_DATA:
        ...
    }
  else if (hdr->destAddr == ANY_DEST)
    switch (hdr->frameType) {
      case M802_11_DATA:
        Mac802_11ProcessFrame(node, M802, msg);
        ...
    }
  ...
}
```

Figure 5.8. Handling a MAC frame from radio interface

type. By using network layer primitives for packet delivery purposes, the simulation code needs no change if different MAC layer protocols are to be tested for performance. It should be emphasized that, other than being a simple payload carrier, IP protocol is not used for any other purposes, such as routing. In this sense, using IP as the protocol packet carrier does not contradict with 'an underlying data link protocol is sufficient' assumption, given in Section 4.1.

Figure 5.9 provides an overall look to the GloMoSim network layer and the API between the neighboring layers. Figure summarizes the API calls that invoke each layer in both directions (i.e., a packet arrival from MAC layer, and a packet departure from transport layer). Basically, network layer processes arriving frames from MAC layer to construct complete IP datagrams. According to the IP protocol number supplied in the datagram (defined IP protocol numbers are found on *include/nwcommon.h*), delivery is made to the correct upper transport layer handling function (currently, UDP or TCP). If the arriving IP datagram is not destined for the host that is processing the packet, a routing action is taken. There are three possibilities in routing an IP packet. The node under consideration may have a registered routing function, meaning that a

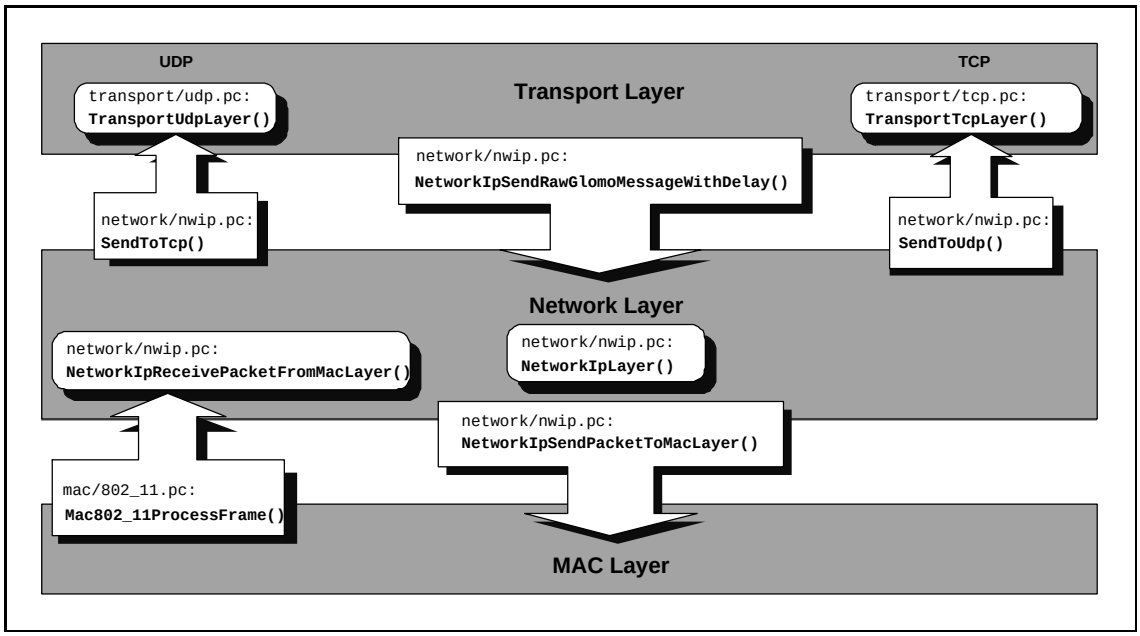


Figure 5.9. GloMoSim network layer and its API to neighboring layers

routing scheme has been selected in simulation configuration file such as DSR, AODV etc. Then this IP packet is given to the registered protocol routing function and the value from this routing function is checked. If routing has been done, no further action is taken. If routing could not be done by the protocol routing function, or the node has no specified routing protocol, the packet is checked to see if it has an IP source route on the IP options field. If a source route is found, the standard IP source routing is done for the packet (i.e., the next hop towards destination is extracted and the packet is forwarded along the destination via this next node). Otherwise, the routing is realized according to the static IP routing table kept at the node (if an entry is found for the destination). The operation of network layer is given in Figure 5.10 as a flowchart.

Design of the implemented protocol elements in the simulation is closely related to the operation of network layer. *Communication Agent* of the protocol acts like a registered routing protocol and intercepts the protocol packets that travels in IP datagrams, if the node is configured to run the proposed ad hoc networking protocol. After intercepting protocol packets at the network layer, they are processed and forwarded to other agents as explained in Chapter 4, without further interaction with existing components of the simulator in upper layers (e.g UDP, TCP, transport).

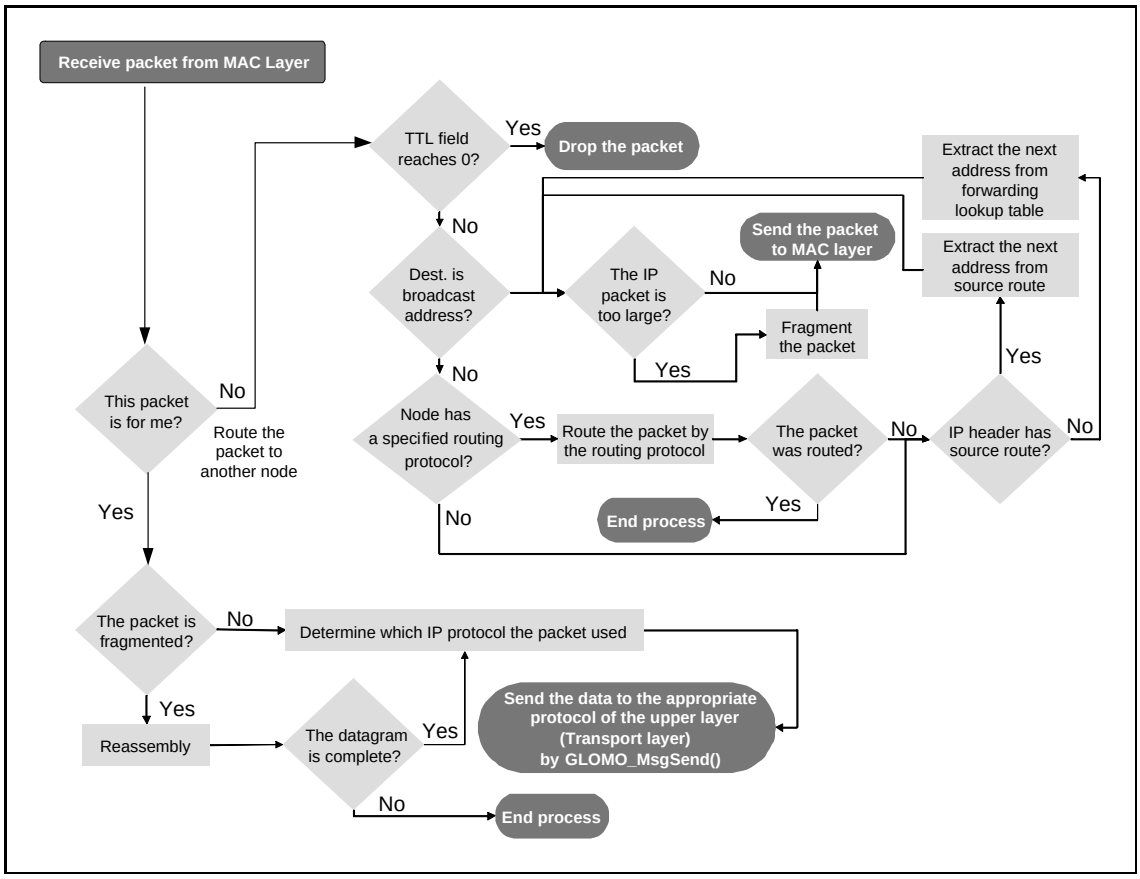


Figure 5.10. GloMoSim network layer operation

5.1.5. Addition of a New Protocol

In order to add a new protocol or layer to the GloMoSim, a main data structure and three group of functions are to be implemented and interfaced to the currently available code skeleton.

First, the data structure for the specific protocol should be implemented. This structure should be designed to hold internal data structures of the protocol, states of the protocol and statistics of the protocol. Upon arrival of a message, the node must be able to determine what to do by only looking at the message and current snapshot of this protocol data structure.

An *initialization function* is necessary to prepare the protocol data structure before the simulation starts (e.g., to reset algorithm counters or to initialize statistics

data structures). This initialization function should be interfaced into the necessary layer handler of the available simulation code so that it is invoked (from *main/glomo.pc*) every time the simulation starts.

Event handling functions are the most important functions that have to be implemented. These group of functions will act upon arrival of various events (e.g protocol timers, lower or upper layer messages) and packets. Algorithms of the added protocol should be implemented in the scope of these functions. In general, two of those functions, *main event handler* and *main packet handler* are to be interfaced into the simulation code of the appropriate layer.

A *finalization function* is also necessary for every newly implemented protocol. Finalization function collects the statistical information kept at the protocol main data structure, processes and outputs them to the main statistic output file (*glomo.stat*). Finalization functions for all protocols are invoked at the end of the simulation in the same way of the initialization functions. Therefore the implemented finalization function must also be interfaced into the available simulation code.

To ease the job of implementing new protocols into the simulation software, empty protocol templates are prepared to be filled in the current distribution of GloMoSim. In the directories of appropriate layers, files starting with '*user*' can be used for this purpose (e.g., *application/user_application.pc*, *network/user_nwip.pc*). The template functions residing in those files have already been interfaced to the necessary places in the simulation software and are very useful in understanding the simulation software mechanics.

5.2. Implementation of Proposed Protocol Simulation

The mechanisms provided in Chapter 4 are implemented and integrated into the simulation software, keeping the proposed structural design as much as possible. The integration of components into the GloMoSim layered stack is illustrated in Figure 5.1 on Page 65. Implemented protocol components are found in the files *network/semarp.pc*

Table 5.2. Results of different XML compression or encoding methods

Method	**File Size (in bytes)**
Original (not processed)	227
gzip compression	207
XMill compression	218
Custom encoding	111

(communication, routing and service agents) and *transport/semasm.pc* (session manager). There are also various application implementations residing in *application* directory with names *sema_cbr_client.pc* and *sema_cbr_server.pc*, *sema_vbr_client.pc* and *sema_vbr_server.pc*, *sema_printer_client.pc* and *sema_printer_server.pc*. Applications are covered in more detail in Section 5.3.

5.2.1. XML Processing

Efficient representation of plain text human readable XML instances is important for the proposed protocol since those instances occupy space in protocol packets. For the purpose, some of possible XML compression and encoding options have been tested during simulation implementation. Tested methods are *gzip* compression [73], *XMill* XML compression [74], and a *custom encoding*. The custom encoding was done by simply assigning one letter for each tag and attribute name (first letter of the original tag and attribute, if not taken) appearing in the XML file (all white spaces are collapsed). Therefore, a pre-determined encoding table is necessary for the purpose.

For a typical host XML instance, tested methods resulted in the file sizes that are summarized in Table 5.2. It is obvious that compression methods performed poor (since instances are quite small in file size) and custom encoding has been found successful when compared to others. Therefore, instances are used encoded by this custom encoding in the simulation implementations. Further discussion on efficient XML representation is found in [75]. The original version and custom encoded version of a typical host instance is given in Figure 5.11 to clarify the encoding process.

ORIGINAL VERSION

```
<mobileHost phyAddress="A3:55:4F:C3:64:B4">
  <battery type="NiMh" maxCapacity="450">200< /battery>
  <security level="high">
    <publicKey id="00" algorithmName="RSA" length="32">110...00< /publicKey>
  < /security>
< /mobileHost>
```

ENCODED VERSION

```
<m p="A3:55:4F:C3:64:B4"><b t="NiMh" m="450">200< /b>
<s l="high"><p id="00" a="RSA" l="32">110...00< /p>< /s>< /m>
```

Figure 5.11. A typical host XML instance and its encoded version

For an efficient simulation code, host and service XML instances are kept in data structures named *hostInstance* and *serviceInstance*. They are defined in *network/semarp.h*. When an XML parsing takes place, the result is kept in a suitable instance structure to save time on future references to the same XML instance (XML instances are usually referenced more than once). Therefore, all other data structures keep XML instances as corresponding host or service structure instances. Whenever necessary (XML version is needed to be put in protocol packets), an XML instance is constructed back from its structure instance.

In the simulation implementation, XML validation and parsing has been done by making use of *Expat* [76], which is an XML parser library written in C. Expat is a stream-oriented parser in which the user registers handlers for various occurrences the parser might find in the XML document (like start tags). Expat library source is compiled with the simulator code and linked against the simulation object files. Expat source files reside in *expat* directory of simulation software.

5.2.2. Communication Agent

Communication Agent of the protocol provides independence from the underlying link layer connectivity primitives. Since the simulation design is made using IP datagram payloads as the protocol packet carrier (See Section 5.1.4 for this discussion), un-

```
                                                    ...from the file network/nwip.pc

static void ProcessPacketForMeFromMac(GlomoNode *node, Message *msg){
  ...
  NetworkIpRemoveIpHeader(node, msg, &srcAddr, &dstAddr, &prior, &IpProto, &ttl);
  switch(IpProto) {
  ...
    case IPPROTO_UDP:
      ipLayer->stats.numPacketsDeliveredToThisNode++;
      ipLayer->stats.deliveredPacketTtlTotal += ttl;
      SendToUdp(node, msg, prior, srcAddr, dstAddr);
    case IPPROTO_DSR:
      RoutingDsrHandleProtocolPacket(node, msg, srcAddr, dstAddr, ttl);
    case IPPROTO_SEMARP:
      RoutingSemarpHandleProtocolPacket(node, msg, srcAddr, dstAddr, ttl);
    default:
      NetworkIpUserHandleProtocolPacket(node, msg, IpProto, srcAddr, dstAddr, ttl);
  }
...
}
```

Figure 5.12. The trap code intercepting SeMA packets from IP payloads

derlying data link layer independency has been achieved already. Therefore in the simulation, communication agent can be seen as the simple trap code in *network/nwip.pc* which forward IP datagrams with protocol number '*IPPROTO_SEMARP*' to the packet handler in *network/semarp.pc*. From this point on, the packet is either in the hands of *routing agent* or *service agent*. The code excerpt for this processing is given in Figure 5.12.

5.2.3. Routing and Service Agents

Simulation of *Routing Agent* and *Service Agent* of the proposed protocol are implemented in *network/semarp.pc* file. The event and packet handler functions residing in the file are invoked from the network layer of the simulator upon arrival of an IP packet with protocol number *IPPROTO_SEMARP* or upon arrival of a non-packet message (See Section 5.1.3 for message types) with one of the events (if destined for routing or service agents) listed in Table 5.3. The events listed in table are presented

Table 5.3. Non-packet messages for protocol components

Name and Explanation of Messages
MSG_NETWORK_SEMARP_EXPAND_LOOKUP Timer message to retry lookup with one step expanded ring
MSG_NETWORK_SEMAAPP_TO_SEMARP_START_LOOKUP Indication from application to start looking for a service (to Service Agent)
MSG_NETWORK_SEMAAPP_TO_SEMARP_OFFER Indication from application to start offering service (to Service Agent)
MSG_NETWORK_SEMASM_TO_SEMARP_ROUTE_PACKET Request for routing a SeMA packet (from Session Manager to Routing Agent)
MSG_TRANSPORT_SEMASM_SERVE_DEPARTURE_QUEUE Indication of processing need at the Session Manager Departure Buffer
MSG_TRANSPORT_SEMASM_TIMEOUT_SESSION Timer message to process a session that is inactive for a long time
MSG_TRANSPORT_SEMARP_TO_SEMASM_DELIVER_PACKET Indication of an unseen data packet (from Routing Agent to Session Manager)
MSG_TRANSPORT_SEMAAPP_TO_SEMASM_START_AUTO_SESSSION Indication from application of a non-interactive session data (to Session Manager)
MSG_TRANSPORT_SEMAAPP_TO_SEMASM_MANUAL_SESSSION Indication from application of an interactive session data (to Session Manager)
MSG_APP_SEMARP_TO_SEMAAPP_SERVICE_STARTED Confirmation from Service Agent of successful service offering start (to application)
MSG_APP_SEMARP_TO_SEMAAPP_FOUND_SERVICE Confirmation from Service Agent of successful discovery of a service instance (to application)
MSG_APP_SEMARP_TO_SEMAAPP_LOOKUP_TIMEOUT Indication from Service Agent of an unsuccessful discovery attempt (to application)
MSG_APP_SEMASM_TO_SEMAAPP_QUEUE_FINISH Indication from Session Manager of a successful session data processing (to application)
MSG_APP_SEMASM_TO_SEMAAPP_HAPPY_END Delivery from Session Manager of a new arriving session data (to application)
MSG_APP_SEMASM_TO_SEMAAPP_TOSERVER_DELIVERY Indication from Session Manager of a successful remote data delivery (to application)

with their names, destination agents and intended meanings. These and other simulation events are defined in *include/structmsg.h* file.

```
                                        ...from the file network/semarp.h

typedef struct glomo_network_semarp_str {
  BOOL semarpStatsEnabled;
  int nextAvailableServiceTIndex;
  int nextAvailableLookupSequenceNumber;
  SEMARP_ServiceTable serviceTable;
  SEMARP_Buffer buffer;
  SEMARP_SessionCache sessionCache;
  SEMARP_LookupCache lookupCache;
  SEMARP_Stats stats;
  SEMARP_HostInstance myInstance;
} GlomoRoutingSemarp;
```

Figure 5.13. Definition of the protocol main data structure

Within the implementation residing in *network/semarp.h* file, protocol main data structure *GlomoRoutingSemarp*, protocol packet structure *SEMARP_Packet*, and statistics data structure *SEMARP_Stats* are defined.

GlomoRoutingSemarp is a composite data structure including many other definitions of structures (e.g., service table, session cache etc.) that are explained in Chapter 4. Definition of this main structure is given in Figure 5.13 and definitions of other structures therein can be found in file *network/semarp.h*.

Protocol packet structure is explained in Section 4.3 and defined as illustrated in Figure 5.14 for the simulations. The constants used in definitions also appear in *network/semarp.h* file.

Statistics related to the particular routing agent and service agent of a node are kept in *SEMARP_Stats* structure. These variables are updated in various parts of the routing and service agent codes, recording the effect of the current state of simulation. The names of the statistics variables and their purposes are listed in Table 5.4. These variables are printed out to the *glomo.stat* statistics output file at the end of the simulation.

```
                                                    ...from the file network/semarp.h

typedef struct {
  SEMARP_PacketType pktType;
  NODE_ADDR clientId;
  long int sessionId;
  long int sequenceNum;
  BOOL lastFlag;
  BOOL lostFlag;
  clocktype twOfServiceInst;
  short routeHopCount;
  char hostsOnRouteXML[SEMARP_MAX_SOURCE_ROUTE_LENGTH][SEMARP_MAX_HOST_XML_LENGTH];
  int payloadSize;
} SEMARP_Packet_Hdr;

typedef struct {
  SEMARP_Packet_Hdr header;
  char payload[SEMARP_MAX_PAYLOAD_SIZE];
} SEMARP_Packet;
```

Figure 5.14. Definition of the protocol packet

In the implementation residing in *network/semarp.pc*, functions that realize following actions are found: agent initialization/finalization, protocol event handling, packet routing, link layer error handling, packet initiating, handling, relaying (announcement, lookup, lookup reply), service table management, session cache management, route healing, XML generating and parsing, and timer management.

5.2.4. Session Manager

The *session manager* implementation of the simulation realizes the functionality explained in Chapter 4 for non-interactive services. Implementation of session manager is found in file *transport/semasm.pc*. Recalling Figure 5.1 on Page 65, routing agent delivers clean and unseen protocol data packets to the session manager.

The event handler of the session manager is responsible from dealing with the non-packet messages in Table 5.3 on Page 5.3 that are destined to the session-manager. Basically, session manager simulation implementation homes functions to fragment and

Table 5.4. Routing and Service Agent statistics variables

Name and Purpose of the Statistics Variable
numServiceLookupsSent: Total number of service lookup packets transmitted
numServiceLookupsReceived: Total number of service lookup requests received
numServiceLookupsAnswered: Total number of service lookup requests replied with a suitable service
numServiceLookupsRelayed: Total number of service lookup packets relayed to other hosts
numServiceLookupRepliesReceived: Total number of service lookup replies received
numServiceLookupRepliesRecorded: Total number of service lookup replies recorded to service table
numServiceAnnouncementsSent: Total number of service announcement packets transmitted
numServiceAnnouncementsReceived: Total number of service announcement received
numServiceAnnouncementsRelayed: Total number of service announcement relayed out
numServiceAnnouncementsRecorded: Total number of service announcements recorded to service table
numSemaPacketsDelivered: Total number of packets delivered to this node
numLinkBreaks: Total number of packets lost due to wireless link unavailability
numHealedRoutes: Total number of packets re-sent with updated routes

packetize the submitted data from the application, trigger service lookup and initiate a non-interactive session. For these purposes, depending on the session type (whether an *inbound*, or *outbound* session), an arrival or departure buffer is maintained. Main data structure for the session manager is a *session table*. The important elements of a session table entry are given in Figure 5.15.

```
                                        ...from the file transport/semasm.h

typedef struct SEMASM_SessionTable_Element {
  long localSessionId;
  long localApplicationId;
  SEMASM_SessionType sessionType;
  SEMASM_Departure_Buffer departureBuffer;
  SEMASM_Arrival_Buffer arrivalBuffer;
  NODE_ADDR remoteAddr;
  int boundServiceTableIndex;
  long remoteSessionId;
  long lastPacketSequenceNo;
  long numOfBytes;
  long numOfPackets;
  long numOfDuplicates;
  long totalNumOfHops;
  BOOL lastPacketArrived;
  BOOL sessionActive;
  BOOL sessionTimedOut;
  ...
  struct SEMASM_SessionTable_Element *next;
} SEMASM_SessionTable_Entry;
```

Figure 5.15. An excerpt from session table definition

Via the statistics variable kept and the finalization function registered, session manager has the ability to provide the following statistics to the user (per session based):

- Number of received protocol packets from the routing agent
- Number of received protocol packets from the routing agent as lost (*lost flag* set)
- Number of received protocol packets from the routing agent as duplicate
- Total number of bytes submitted by the application for delivery
- Total number of bytes received for the session
- Session duration
- Average throughput
- Minimum, maximum and average end-to-end delay
- Session status (termination not received, incomplete, complete etc.)

5.3. Implemented Applications

To be used in the experiments for performance evaluation, applications that use the proposed protocol are developed and implemented into GloMoSim. The experiments designed with these applications are explained in Section 6.4. The realized implementations fall into *printing*, *CBR class*, and *VBR class* application categories.

5.3.1. Printing Application

Printing application implementation consists of two components. They are printer server (residing on *sema_printer_server.pc* file in *application* directory) and printer client (residing on *sema_printer_client.pc* file in *application* directory).

Functionally, printer server offers the printing service that is specified in the simulation application configuration input file (*app.conf*) via the 'SEMA OFFER' directive (See Appendix B for a detailed example). The server then accepts sessions containing printing requests of clients that successfully discovered this printing service on the host. Arrival of the complete non-interactive session data is assumed to be a successful printing of a file.

Printer client uses the primitives provided to discover a printing service specified in the application configuration input file. The required printer specification is given with 'SEMA ASK' directive (as a service XML instance) and within this directive, there exists also the name of the file that is to be printed. The filename is given relative to the *files/printer* directory and the non-interactive session data size is determined according to this given file to be printed. No retransmission facility is implemented in the printer client application.

5.3.2. CBR Class Applications

CBR (Constant Bit Rate) applications offer a constant flow of data to the network of concern. CBR class of applications are implemented with the idea of 'more

applications belonging to this class may be integrated into the simulation in the future' in mind. As a generic CBR application, current implementation allows users to specify application session duration (in terms of time or item count), item sizes (in bytes) and item inter-departure times. CBR application, similar to the printing application, consists of a service offering CBR server (*sema_cbr_server.pc*) and a service initiating CBR client (*sema_cbr_client.pc*).

As an application instance belonging to CBR application class, *voice over SeMA* (VoSeMA) has also been implemented. Using this application, determined pairs of hosts may run voice sessions that are either encoded in G.723.1 (high quality) or G.711 (medium quality). More information on voice encoding using these and other formats are given in [77]. While converting codec rates to CBR units (payload size, packet per second etc.), information provided in [78] is used and *Voice Activity Detection* (VAD) has not been taken into account. These rate information can be found in file *sema_cbr_client.h*. In the configuration file, VoSeMA client specifies which host to call and the codec to be used. For a probable voice session to take place, the intended host should have announced a VoSeMA service and the client should have discovered the VoSeMA service on that intended host.

5.3.3. VBR Class Applications

VBR (Variable Bit Rate) applications offer data flow to the network in a time-varying manner. Similar to CBR application implementation, VBR application implementation allows integration of different applications belonging to this class. Varying data flow to the network at the VBR client implementation (residing in *sema_vbr_client.pc* file) is achieved by setting application timers for each successive packet. VBR server (residing in *sema_vbr_server.pc* file) is only responsible from offering the VBR service and keeping statistics about the available remote parties.

The implemented application instance belonging to VBR class is a *video streaming* service. Using this application, determined pairs of hosts may benefit from a video stream service which actually transmit MPEG-4 encoded streams from Star Wars IV

movie. The traces are from Telecommunication Networks Group of Technical University of Berlin. Details of traces and encoding process are explained in [79]. The trace file *Terse_StarWarsIV_10_14_18.dat* is used to generate video traffic and can be found in *files/video* directory. For a probable video streaming session to take place, the intended host should have announced a *videostream* service and the client should have discovered the service on that intended host. In the application configuration file, video service client specifies the duration of streaming (up to sixty minutes of video data is available in trace files), name of the stream data (i.e., movie) and the encoding used (i.e., MPEG-4).

5.4. Use of Simulator

GloMoSim compilation is assisted by a 'Makefile' residing in *main* directory of the source hierarchy. After compilation (parsec and generic C-compiler phases), simulation binary, *glomosim*, is found in bin directory. Simulation binary is invoked with one parameter, which is the simulation configuration input file. By default, the configuration file is named *config.in* and found in *bin* folder.

The *config.in* file homes directives for simulation related settings. They are classified as:

- General Simulation Parameters (number of nodes, terrain size, node placement strategy)
- Mobility (model used and its parameters)
- Radio and Propagation Model (frequency, fading power thresholds etc.)
- MAC Protocol
- Routing Protocol
- Transport Protocol
- Application (name of external application configuration file)
- Statistics and GUI options (enabling/disabling specific layer statistics)

An excerpt from a sample *config.in* file is given in Figure 5.16. The possible external

```
SEED 1
NUMBER-OF-NODES 25

...

APP-CONFIG-FILE ./input/app-heavy.conf
SIMULATION-TIME 5M
NODE-PLACEMENT GRID
GRID-UNIT 180
TERRAIN-DIMENSIONS (925, 925)

...

MOBILITY RANDOM-WAYPOINT
MOBILITY-WP-PAUSE 0S
MOBILITY-WP-MIN-SPEED 3
MOBILITY-WP-MAX-SPEED 5
MOBILITY-POSITION-GRANULARITY 0.5

...

PROPAGATION-LIMIT -110.0
PROPAGATION-PATHLOSS TWO-RAY
NOISE-FIGURE 10.0
TEMPARATURE 290.0
RADIO-TYPE RADIO-ACCNOISE
RADIO-FREQUENCY 2.4e9
RADIO-BANDWIDTH 2000000
RADIO-RX-TYPE SNR-BOUNDED
RADIO-TX-POWER 8.0
PROPAGATION-FADING-MODEL RICIAN
RICIAN-K-FACTOR 5

...

MAC-PROTOCOL 802.11
NETWORK-PROTOCOL IP
NETWORK-OUTPUT-QUEUE-SIZE-PER-PRIORITY 500
ROUTING-PROTOCOL SEMARP

...

APPLICATION-STATISTICS YES
TCP-STATISTICS NO
UDP-STATISTICS NO
SEMASM-STATISTICS YES
SEMARP-STATISTICS YES
ROUTING-STATISTICS NO
NETWORK-LAYER-STATISTICS YES
MAC-LAYER-STATISTICS NO
RADIO-LAYER-STATISTICS NO
MOBILITY-STATISTICS NO

...

GUI-OPTION NO
```

Figure 5.16. An excerpt from a sample *config.in* file

files that may be found in a configuration file are *node placement file* (if each node's initial position will be provided externally), *mobility trace file* (if each node's mobility traces will be provided externally), *path loss trace file* (if path loss data will be provided externally), and *application configuration file*.

Application configuration file includes details about the applications that will be set during the simulation. It homes directives specifying node number, application start time, duration etc. The format adopted for the applications explained in Section 5.3 is summarized in the following two paragraphs. Further explanations of the format of the application configuration files can be found as comments in respective application configuration files.

The directive to offer a service from a given node is 'SEMA OFFER' and use of this directive is as follows:

`SEMA OFFER <provider> <start time> <valid until> <serviceXML>`

where,

<provider>:	is the node address of the node on which the service will be announced
<start time>:	is the time at which the service will be announced
<valid for>:	is the time interval during which the service will stay available (0 meaning till the end of the simulation)
<serviceXML>:	is a valid and well-formed (conforming to *Service Schema* documents) service XML instance to be announced, terminated with a '|' (pipe) character

The directive to look for and use a service from a given node is 'SEMA ASK' and use of this directive is as follows:

SEMA ASK <client> <start time> <serviceXML>

where,

<client>:	is the node address of the node on which the service is requested and to be used
<start time>:	is the time the service is requested
<serviceXML>:	is a valid and well-formed (conforming to *Service Schema* documents) service XML instance that is requested to be used, terminated with a '|' (pipe) character

For the both directives (to offer and ask for a service), other application specific details are given embedded into the XML instance of that specific service. A detailed application configuration file is provided in Appendix B.

After the simulation execution, layer statistics are written to file *glomo.stat* per node basis. Further analysis on the results are obtained by processing this file residing in *bin* directory.

6. EXPERIMENTS AND RESULTS

In this chapter, we explain our motivation for the realized simulation experiments, the parameters that have effect on the performance and the ones we populated as factors. After defining some performance metrics, details of experiments are given and finally the results are presented.

6.1. Motivation

Our aim in doing experiments is basically to prove the concept of the service centric approach to be working. If the proof of concept is found acceptable, the experiments will be used as the first design feedback for the choices made on the algorithms and protocol parameter values. Proposed protocol has many aspects to evaluate and among those, we try to focus on the ones that we believe to be important.

6.2. Parameters

The parameters that have effect on the outcome of the experiments are classified into the following two categories as suggested in [80].

6.2.1. System Parameters

System parameters that are listed below have both hardware and software related items and found to be effective on the performance of the proposed protocol:

1. Available radio bandwidth
2. Operation frequency of the radio equipment
3. Radio power (TX/RX power, sensing thresholds etc.)
4. Antenna gain of the radio equipment
5. Selection of the underlying data link protocol
6. Noise characteristics of the environment (thermal, wireless interference etc.)

7. Target terrain structure (dimensions, rural, urban, suburban etc.)
8. Host distribution on the terrain (number, placement etc.)
9. Mobility characteristics of the hosts (behavior, speed etc.)
10. Maximum permitted source route length
11. Maximum permitted XML instance sizes (for hosts and services)
12. Service lookup expanding ring properties (start size, step size, timeout etc.)
13. Service announcement repetition
14. Application retries on service discovery failures
15. Application retransmission facility
16. Availability of services for a given service request

6.2.2. Workload Parameters

Workload parameters are related with the offered load to the proposed system and listed below:

1. Number of applications offering a service
2. Number of applications asking for a service
3. Number of concurrent sessions taking place on network
4. Types of services (CBR, printing, VoSeMA, videoStream etc.)
5. Service parameters (CBR packet size and inter-departure times, printed file size, VoSeMA codec type and duration etc.)

6.3. Performance Metrics

This section provides a description of the performance metrics considered in this study. Some of the listed metrics are for future references and not used in the presentation of the results. Understanding these metrics are vital to correctly evaluate the results presented in Section 6.5.

The *total number of service instances recorded* is a metric giving the total number of entries found in service tables of all nodes, at the end of the simulation duration.

This number gives two insights. First one is how successful the announcement relaying and recording mechanism was in limiting the number of alternatives recorded locally in service tables. Second is how many route alternatives, on average, a node may see to be used in optimization of routes in case of route loss. This metric should be understood as a cost to be paid in bandwidth (announcement packets on the network) and memory space (recorded entries on the service table).

The *total number of link breaks* is another metric and gives the number of times the communication agents were unsuccessful in finding the next hop specified in the source route generated by routing agents. This metric is useful in evaluating the freshness of the source routes generated in the network. Every link break is an additional cost that causes additional packets to be transmitted in the network (either via route healing or flooding).

The *packet delivery ratio* (PDR) is a rather frequently used metric, stating the ratio of the packets that made their ways to destination over all generated packets. It is an overall performance evaluation metric, evaluating the protocol as a tool to transmit all generated packets to their respective destinations.

Average number of hops is again a frequently used metric in routing. But for our protocol, this metric also tells about the performance of the service discovery algorithm in finding the nearest service possible, since routing is inherently related to this property.

Number of failed service binding attempts metric summarizes the performance of the architecture in finding at least one suitable service. However, a service binding failure can not be only related to the operation of our protocol (i.e., a suitable service may be physically unreachable because of the current topology).

Service discovery latency gives the performance of the lookup mechanism in terms of amount of delay incurred. It is measured from the time the application asked for the service to the time service agent returned a suitable discovered instance.

Throughput and *average end-to-end delay* for successfully completed sessions are classical performance metrics giving the amount of available bandwidth and suffered delay for applications.

Number of successfully completed sessions is a metric that may complement PDR, in the sense that the distribution of packet delivery ratio among sessions is evaluated.

Ratio of broadcast packets to unicast packets is a metric that gives an insight on the amount of contention generated in the wireless broadcast network by announcement packets and flooded data packets.

Ratio of control packets to data packets is a classical metric that gives the portion of the consumed bandwidth to make the protocol work.

6.4. Experiments

Experiments are created with the intention of generating a realistic scenario of ad hoc networking. For this purpose, the target environment is selected as a campus field with different characteristic user applications running. The fixed parameters and populated factors are chosen to reflect an acceptable real target environment.

In experiment design, the most of the parameters of Section 6.2.1 have been fixed at values as pessimist as possible in order to have worst case results in that sense. The fixed parameters for the simulations are given in Table 6.1. In the table, two of the parameters that are listed as fixed are not actually separate parameters but appears fixed as a result of others. First is the *approximate wireless transmission range* and this parameter value is fixed approximately at (a time varying value dependent on various simulation conditions) 250 meters as a result of fixed transmission power, reception power threshold, fading and pathloss model. The second such parameter is the *maximum SeMA packet payload size* which is actually determined by the underlying MAC protocol frame size (which is IEEE 802.11 for our discussion), and size of SeMA packet header (heavily dependent on allowed XML instance sizes).

Table 6.1. Fixed parameters of the simulations

Parameter Name	**Fixed Value**
Available radio bandwidth	2 Mb/s
Operation frequency of the radio equipment	2.4 GHz
Radio transmission power	15 dBm
Radio reception sensitivity	-81 dBm
Radio antenna gain	0 dBm
Underlying MAC protocol	IEEE 802.11
Noise characteristics of the environment	Thermal Noise Aware / $290°K$
Terrain Pathloss Model	Two-Ray
Terrain Fading Model	Rician with *K factor* 5
Approximate wireless transmission range	250 meters
Host placement strategy on terrain	Variant of Uniform Distribution
Maximum permitted source route length	10 hops
Maximum permitted host XML instance size	50 bytes
Maximum permitted service XML instance size	200 bytes
Maximum SeMA packet payload size	1488 bytes
Service lookup expanding ring start diameter	1 hop
Service lookup expanding ring step diameter	1 hop
Service lookup expanding ring retry limit	3 times
Service lookup expanding ring timeout	300 ms
Application retries on service discovery failures	3 times with 1 sec. backoff
Application retransmission facility	None
Simulation time	5 min

In the target campus environment, there are high number of low mobile pedestrians, some bicycle riding mobile users and few low speed vehicles. The distribution of such mobiles with characteristic speeds are given in Table 6.2 and lead us to the average mobile user speed used in the base problem, which is 3 m/s. Mobility of the users are imitated by making use of random waypoint mobility model included in the simulator (see Section 2.3.1 for details of the model). Pause times of mobiles are selected to be 30 seconds, which resembles the settling of a user upon reaching its intended destination. During five minutes of simulation time, mobiles find enough time to pause and continue their movements. The results of random waypoint mobile network are compared with

Table 6.2. Distribution of mobile users and their speeds

Mobile Class and Avg. Speed	**Distribution**
Pedestrian (2 m/s)	% 75
Bicycle Rider (4 m/s)	% 15
Vehicle (10 m/s $\cong$ 36 km/h)	% 10
Chosen Overall Average Speed	3 m/s

no mobility (fixed network) alternative.

The scenarios under consideration take place on rectangle or square shaped terrains where mobile hosts are placed using a variant of uniform placement. The uniform placement of nodes (i.e., each node placed on a suitable grid intersection in a deterministic manner) is modified as follows. Simulation area is divided into a number of square shaped cells. Except for the 25 host scenario, cells are placed to form a [10 $cell * \frac{n}{10}$ $cell$] structure, where n is the number of hosts in the simulation. Within each cell, a node is placed randomly. Cell dimensions are selected carefully, not to cause a partitioned network at the beginning, and not to cause a highly connected network. For this reason, the *node density*, ρ, is defined to be the average number of wireless transmission neighbors of a given host. Assuming ideal circular transmission shapes, ρ is given as:

$$\rho = \frac{\pi r^2}{a^2} \tag{6.1}$$

where r is the wireless transmission range of the host, and a is the dimension of a cell. For the base scenario, 50 hosts are placed to a terrain of (1850 m * 925 m) with each cell size being 185 m. The node density, ρ, is approximately 5.7 for this base scenario. In order to eliminate the effect of workload increase and only observe the scalability, the variations on the terrain size is done with keeping the ρ approximately same. The variation of terrain dimensions for the simulations are illustrated in Figure 6.1 with their cell structures. To avoid a corridor-like simulation area, which may cause unrealistic moves of mobiles, [10 $cell * \frac{n}{10}$ $cell$] structure is not used for the 25 host case. Instead, the terrain is selected to be a square shaped area of (925 m * 925m), keeping ρ approximately in the same range.

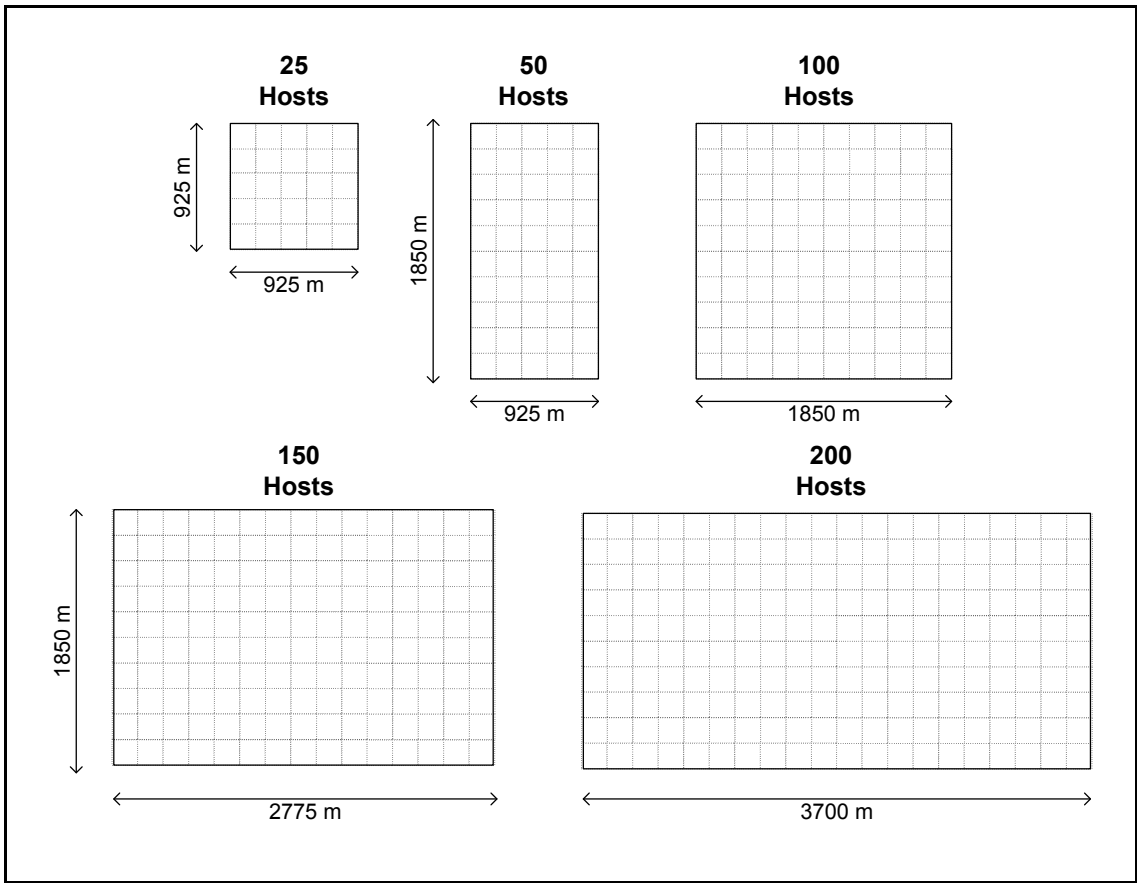


Figure 6.1. Simulation terrain dimensions and placement cells

Applications that are implemented for simulations may be used in many combinations to introduce workload to the system. There are five classes of services to be selected with various internal parameters. They are *printing*, *generic CBR*, *generic VBR*, *VoSeMA (voice over SeMA)*, and *video streaming*. To ease the factorization of workload parameters, two application scenarios are designed to be a reasonable mixture of these applications, trying to reflect the real traffic of a campus environment. These two application scenarios and their parameter details are given in Table 6.3.

For the application scenarios given in Table 6.3, hosts that provide and ask services are randomly selected and do not change from one test to another. However, as the uniform placement strategy places nodes differently for different terrain sizes, the starting topology for applications change unavoidably. The request times of services are again randomly distributed over the simulation duration. Same application configuration files are used for {50, 100, 150, 200} host simulations. Application con-

Table 6.3. Application scenarios used in simulations

Application Parameter	**Light Application Scenario**	**Heavy Application Scenario**
Number of Printers Available	5 identical	10 identical
Number of Hosts Asking Printer Service and Printed Document Type	6 hosts (RFC / 6KB) 3 hosts (User Manual / 56KB) 1 host (Book / 152KB) appx. 240 SeMA Packets a total of 356KB	12 hosts (RFC / 6KB) 6 hosts (User Manual / 56KB) 2 host (Book / 152KB) appx. 480 SeMA Packets a total of 712KB
Number of VoSeMA calls (in pairs)	3 pairs 20 seconds each G.723.1 codec Bit rate 5.3Kb/s	6 pairs 20 seconds each G.723.1 codec Bit rate 5.3Kb/s
Number of Video Stream Sessions	1 host 60 seconds MPEG-4 encoded Star Wars IV Bit rate 54Kb/s (mean)	2 hosts 60 seconds MPEG-4 encoded Star Wars IV Bit rate 54Kb/s (mean)

figuration files for 25 host network is re-edited to be consistent with node numbering, since applications were distributed among hosts of base scenario (50 hosts).

In the application configurations, printers are identical in attributes, therefore any printer requesting application may print to any of the available printers. Voice and video sessions however, take place between two pre-determined (randomly) hosts of the ad hoc network. This is realized by using an extra attribute called *calee* for voice applications and *target* for video streaming applications. The values of the attributes are set to match their remote party host identifiers, so that the client can only bind to the service on its intended provider.

For the base problem, validity duration for applications are selected to be 100 seconds, which means that the applications are re-announced three times for a simulation duration of five minutes. The input file for light application workload is given in Appendix B for the base problem.

6.5. Results

In this section we present the results of the various simulation runs, classified according to the factors whose effects are of interest. The presented results are statistical averages taken from ten different simulation trials. Most of the results are plotted with 90 per cent confidence intervals (as error bars) that are calculated for the unknown mean. Note that some of the intervals are smaller than the symbols used to represent data points, which can be commented as 'the plotted mean is very close to the unknown mean'.

6.5.1. Service Announcements Recorded

In Figure 6.2, the total number of service instances that are recorded to local service tables are given for five different network configurations under light application workload. The results are presented for both a static host (hosts stay at their initial places) and mobile host network (random waypoint with 3 m/s average mobile speed).

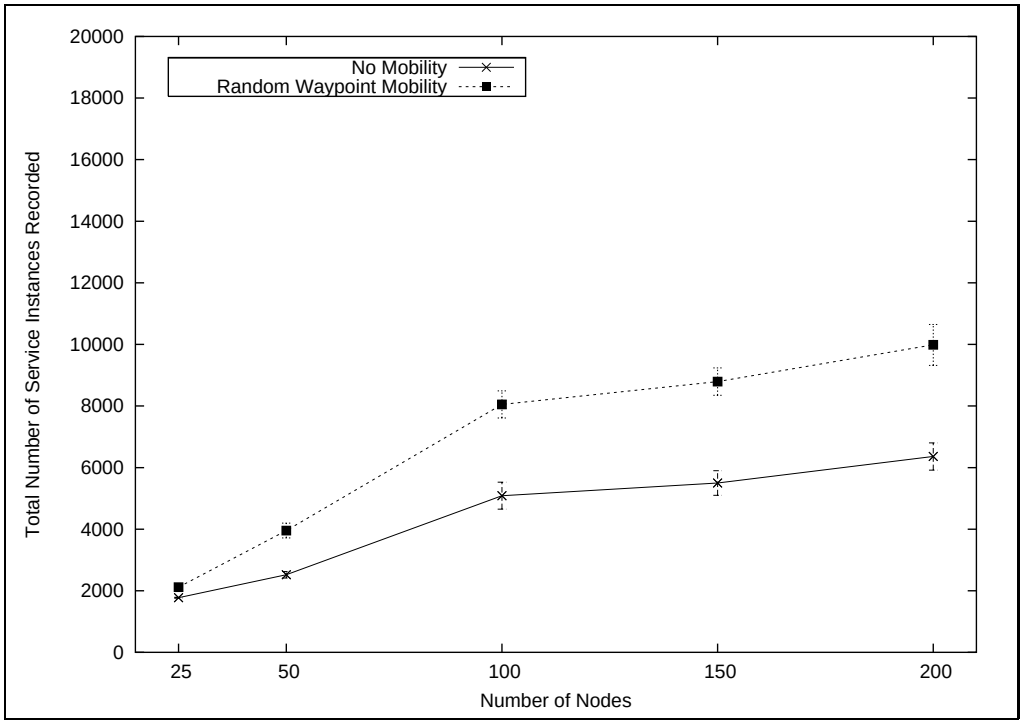


Figure 6.2. Total number of service instances recorded under light application scenario

As the size of the network grows, two factors affect the increase of the number of total service instances recorded. The first and obvious is the increase of the number of nodes. As more hosts are added, those new hosts record the announcements they have heard. The second is related to the number of different alternative paths that a service announcement may be heard from. As we have more nodes on a larger terrain, probability of hearing an announcement from a far node increases, which creates more alternative entries for a constant number of announced services.

For 100, 150, and 200 node networks, the increase in the number of recorded instances slows down and starts to saturate, because of the entry limit per service instance, explained in Section 4.4.2.2. This limit is selected to be five in the simulation runs and puts upper limit on the space used for service table.

It is also clearly seen that mobility, causing nodes to be in different places during the simulation, helps nodes to hear more service announcement alternatives, thus increasing the number of service instances recorded.

Same trends are observed for the heavy application scenario as illustrated in Figure 6.3. Total number of announcements recorded is higher since heavy application scenario includes more (literally double of light application scenario) services to be announced.

6.5.2. Application Oriented Performance

In this section, performance of the protocol is evaluated from the view of printing applications taking place in heavy application scenario. There are ten printers and twenty clients try to get printing service from those printers.

In Figure 6.4, average end-to-end delay suffered by the printing service sessions are presented. As more nodes are added to a larger terrain, suffered average end-to-end delay increases, since services are fetched from more hop incurring paths. Compared to the static scenario, end-to-end delay is worse for the mobile scenario, since number

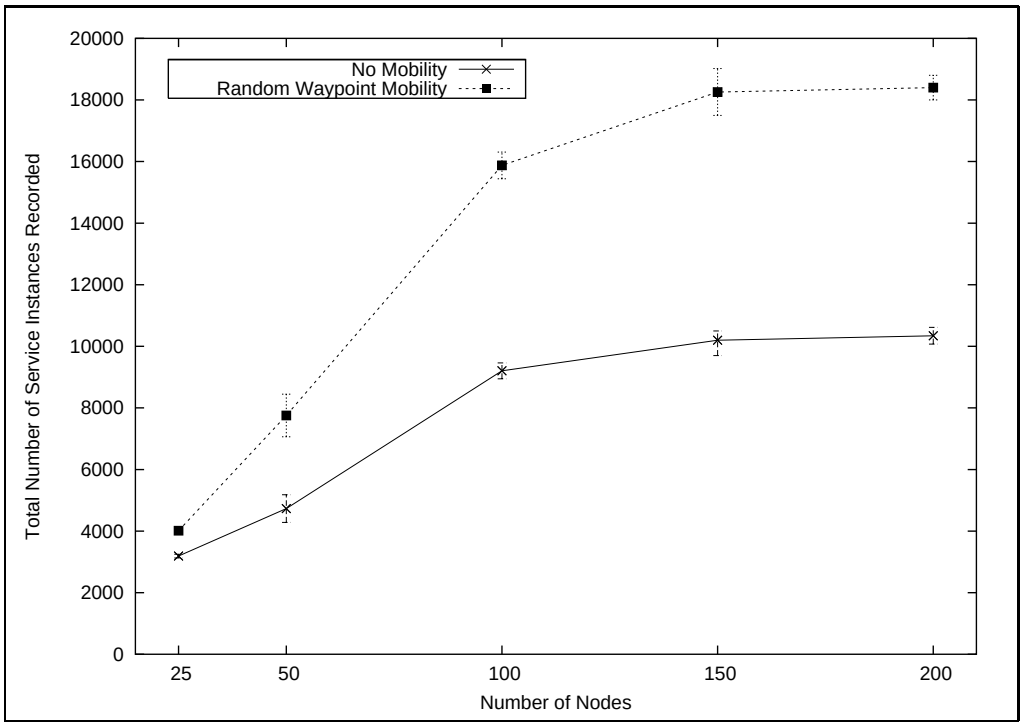


Figure 6.3. Total number of recorded service instances under heavy application scenario

of hops a session is carried over changes as mobiles move. Additionally, route loss and healing takes extra time, resulting end-to-end delay time increase.

Total number of printing sessions completed intact are given in Figure 6.5. Analysis is done for the heavy application scenario, where twenty printing jobs are submitted to various printers. For the static network configuration, most of the printing jobs are completed. Worst case, two or three printing jobs are missed due to either an unsuccessful service discovery (no physically reachable service is found for the topological layout) or lost packets as a result of a temporal contention (e.g., during an announcement repetition) on the network. However, with the introduction of mobility, some more printing sessions become incomplete. This is due to the frequent route changes and possible route losses resulted from mobility of the nodes. For both static and mobile scenarios, a small network (i.e., 25-node) has advantages because of its small terrain size in completing sessions. Average number of hops of sessions are low in smaller network configurations.

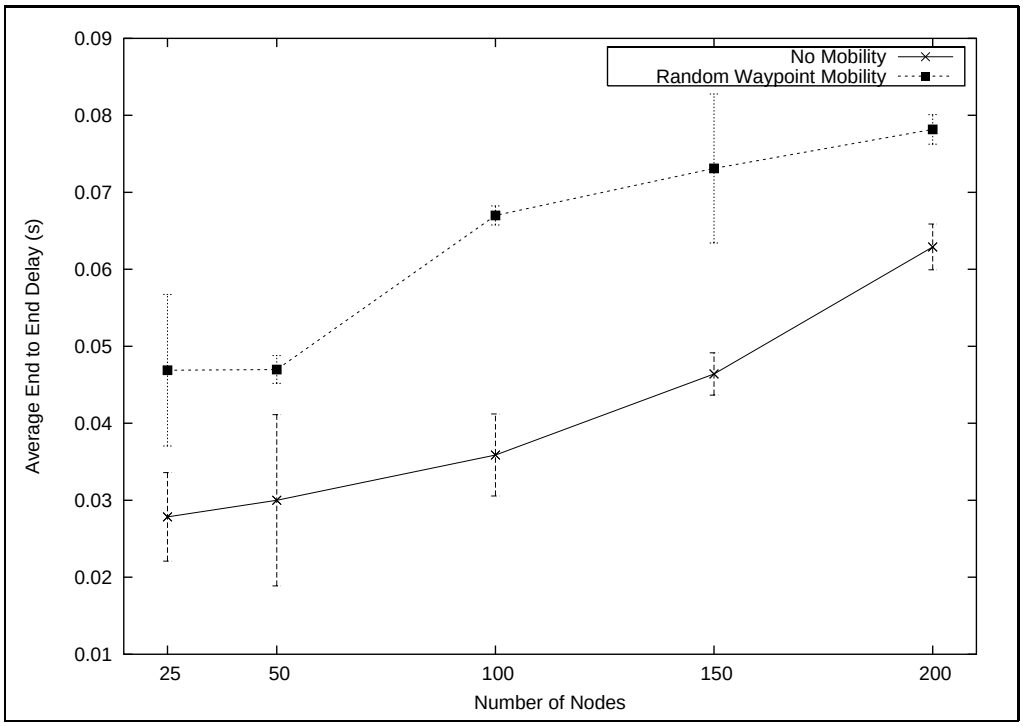


Figure 6.4. Average end-to-end delay for printing applications of heavy application scenario

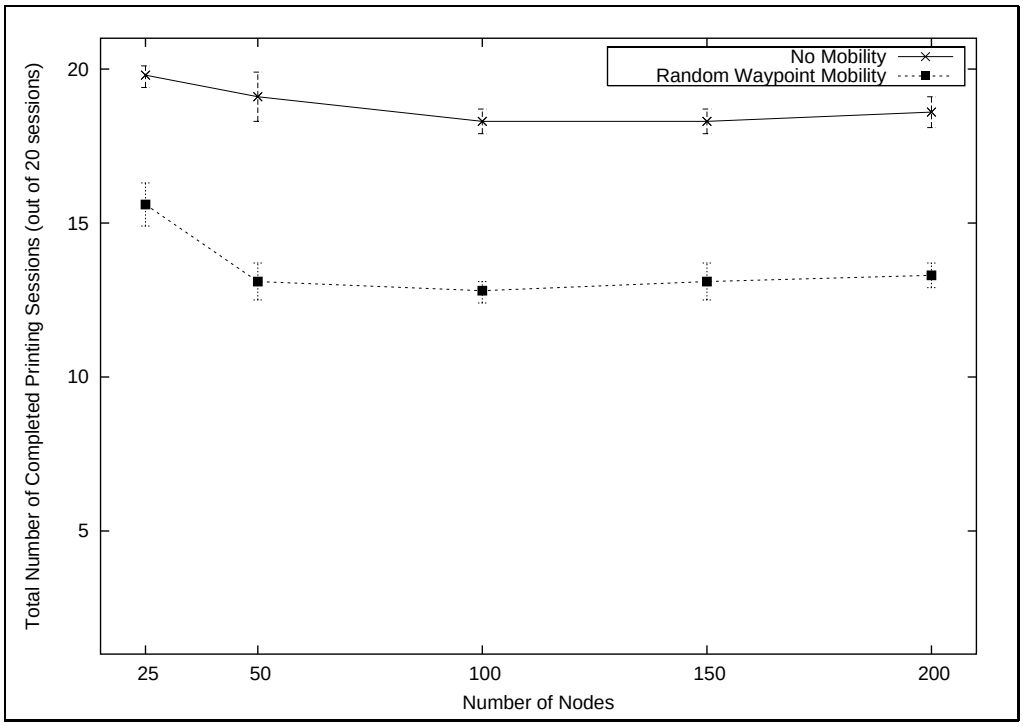


Figure 6.5. Number of completed printing sessions for heavy application scenario

Table 6.4. Percentage of services discovered via lookup and average discovery latencies for heavy application scenario

Number of Hops	**Percentage of Discovered Services via Lookup (per cent)**	**Average Service Discovery Latency (s)**
25	2	1.09
50	0.7	0.17
100	4	1.07
150	0	0
200	2	0.13

6.5.3. Service Discovery Latency

The amount of time passed between the start of the service fetch process and the time on which a suitable service instance is returned to the application is defined as the service discovery latency. The latency for locally discovered services are zero. Rest of the discoveries are realized via service lookup procedure and their percentage (comparing them to locally discovered ones) and latency values are presented in Table 6.4. The results presented in the table are for 25, 50, 100, 150 and 200 node networks using random waypoint mobility under heavy application scenario. From all bindings realized, most of the services are discovered from respective service agents local service tables, thus introducing no extra delay. Rest of the bindings are realized with the help of service lookup mechanism, and produced the average delays provided in the table.

This result is expected since no service announcement scope limiting mechanisms are used in the protocol. Only need for lookup is caused by mobility. If a node has missed an announcement because of being somewhere out of the transmission range of others, a lookup is necessary for that node to bind to that missed service. From the results, it is observed that this has occurred at most 4 per cent of the time.

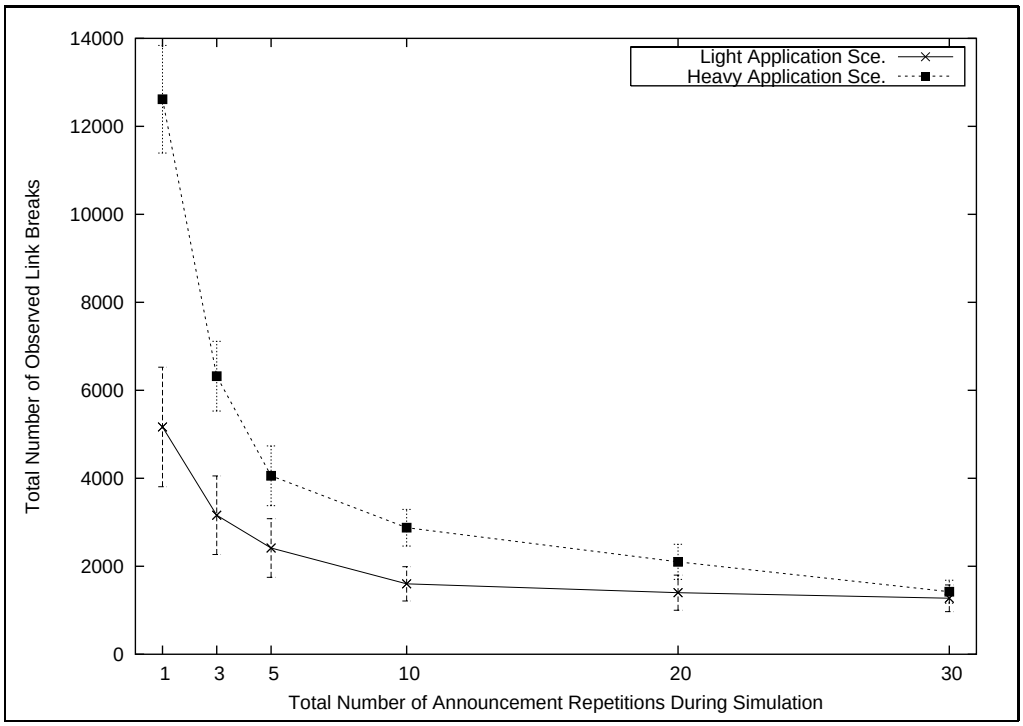


Figure 6.6. Total number of link breaks as services announced more frequently

6.5.4. Effect of Service Announcement Repetitions

For the base problem (50-nodes with random waypoint mobility), effect of the number of service announcement repetitions during the simulation is inspected for heavy and light application scenarios. Number of repetitions are given for the all simulation duration. For example, ten repetitions mean that the services are announced at seconds 0, 30, 60, 90, 120, 150, 180, 210, 240, 270 during a simulation of five minutes. From Figure 6.6, importance of the number of times that the services are re-announced has an important effect on the overall performance of the protocol, in the sense that service table entries are fresh and has valid source routes. Observed link breaks (unavailability of the next hop indicated in the source route) in the network decrease exponentially as services are re-announced more frequently. This behavior is expected, since sessions make use of fresh entries having routes that are not broken yet because of mobility.

Figure 6.7 further supports this idea where important improvements on the packet delivery ratio are observed both for light and heavy application scenarios. The only

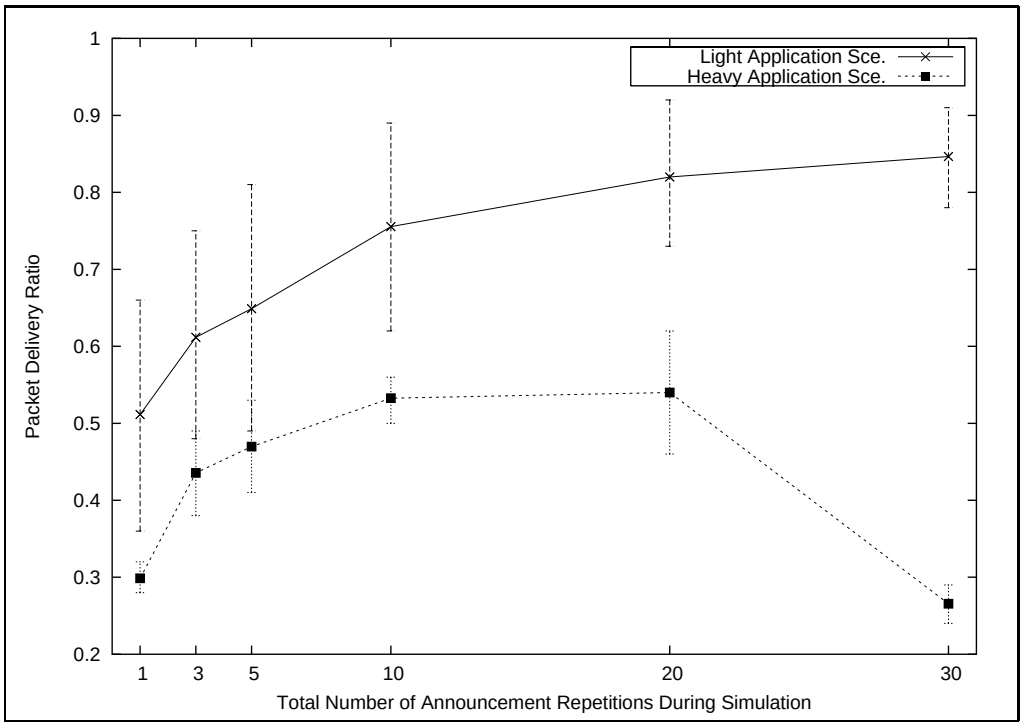


Figure 6.7. Packet delivery ratio in the network as services announced more frequently

exception is seen on the heavy application scenario running network with 200-nodes, when each of the twenty-eight services are announced thirty times during the simulation. The sharp break in PDR increase trend indicates the threshold where the medium contention generated by the broadcast service announcement packets start affecting the transmission of unicast data packets. Therefore, depending on the operation point of the network (in terms of offered load), there exists an upper limit beyond which more service announcement repetitions are useless and decrease PDR success.

The advantage of announcement repetitions has its price paid in terms of service announcement traffic generated. The increase in the amount of announcement traffic can be observed from the total number of instances recorded at hosts. As seen from Figure 6.8, number of recorded service instances increase almost linearly with number of service announcement repetitions. This situation is worse for the heavy application scenario, requiring more announcements to be relayed in the network.

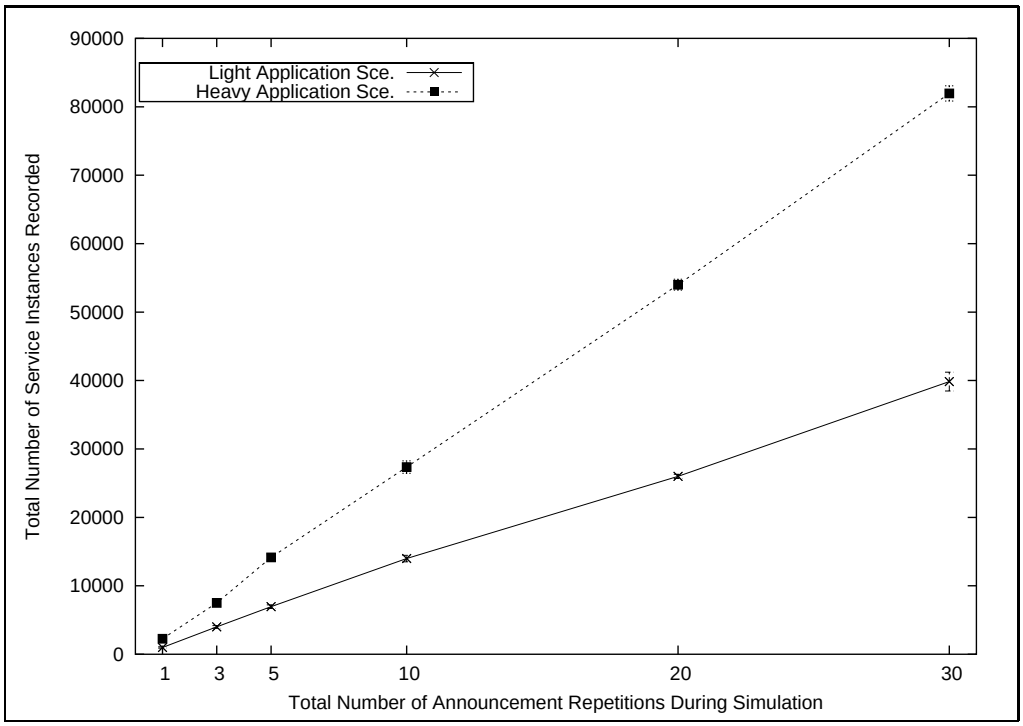


Figure 6.8. Total number of service instances recorded as services announced more frequently

6.5.5. Effect of Average Mobile Speed

In this subsection, effect of average mobile speed to the network performance is evaluated. The analysis is made for the base problem (50-nodes with random waypoint mobility) and average speed of mobiles are increased to be 4, 5, 6 and 7 m/s. Results are presented both for light and heavy application scenarios.

It is observed from Figure 6.9 that increasing average mobile speed up to 5 m/s helps mobiles to converge to the alternative routes faster and the new selected route works some time before it is broken again. For speeds of 5 m/s and higher, nodes move faster and routes get invalid faster than the newly switched route could improve the packet routing performance. Therefore the improvement trend on link breaks starts to disappear for mobiles of speeds 5 m/s or greater. The terrain properties are very important when evaluating the results of mobility related results since node's behaviors on terrain boundaries gain more importance as mobile speeds increase.

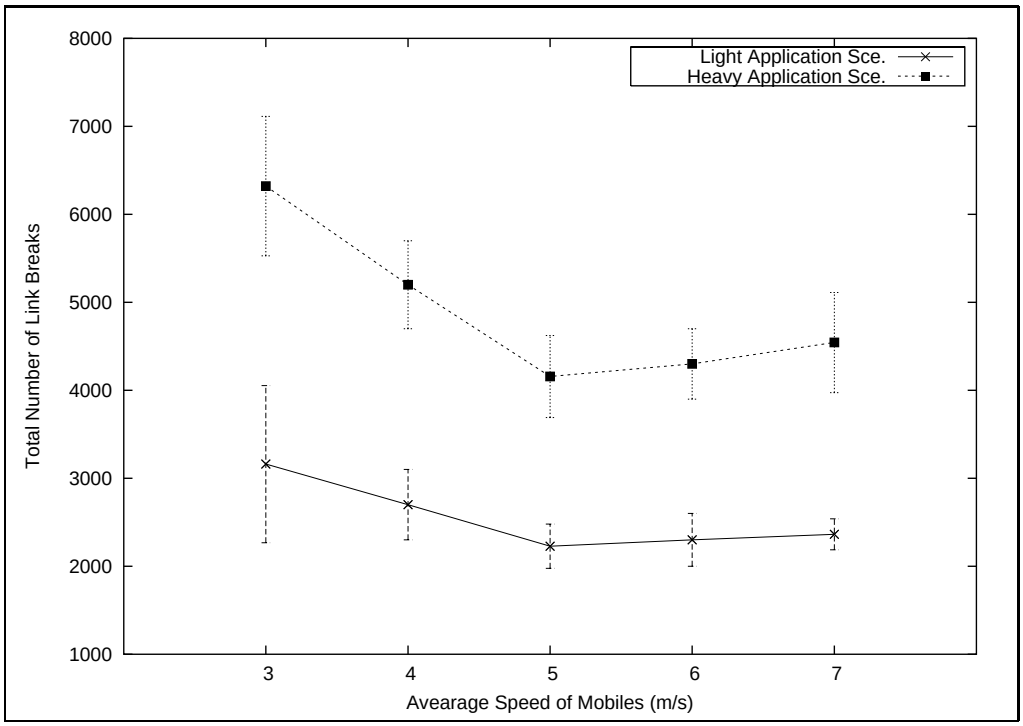


Figure 6.9. Total number of link breaks as average mobile speed changes

As nodes move faster, number of recorded service announcement instances decrease constantly (both for heavy and light application scenarios) as seen from Figure 6.10. This behavior can be related to two reasons. First, the probability that a node may receive an announcement from all different possible paths decrease because nodes change places faster. Second reason is related to the underlying radio and propagation model. Packet error rates increase as nodes move faster. Therefore, comparably more announcements are lost on the way as node speeds are increased.

6.5.6. Routing Agent Performance

In order to come up with a comparable evaluation of performance of the routing mechanism, a frequently used CBR performance evaluation scenario is adapted for our protocol. For our case, the configuration is similar to the experiments defined in [81]. Presented results are for a 100-host network with 40 CBR sessions (between 80 hosts of the network). In each session, CBR client node sends another 512 byte item at every other second to its remote party. In order to avoid unnecessary medium contentions, packet transmission periods of those 40 sessions are adjusted to be one milliseconds

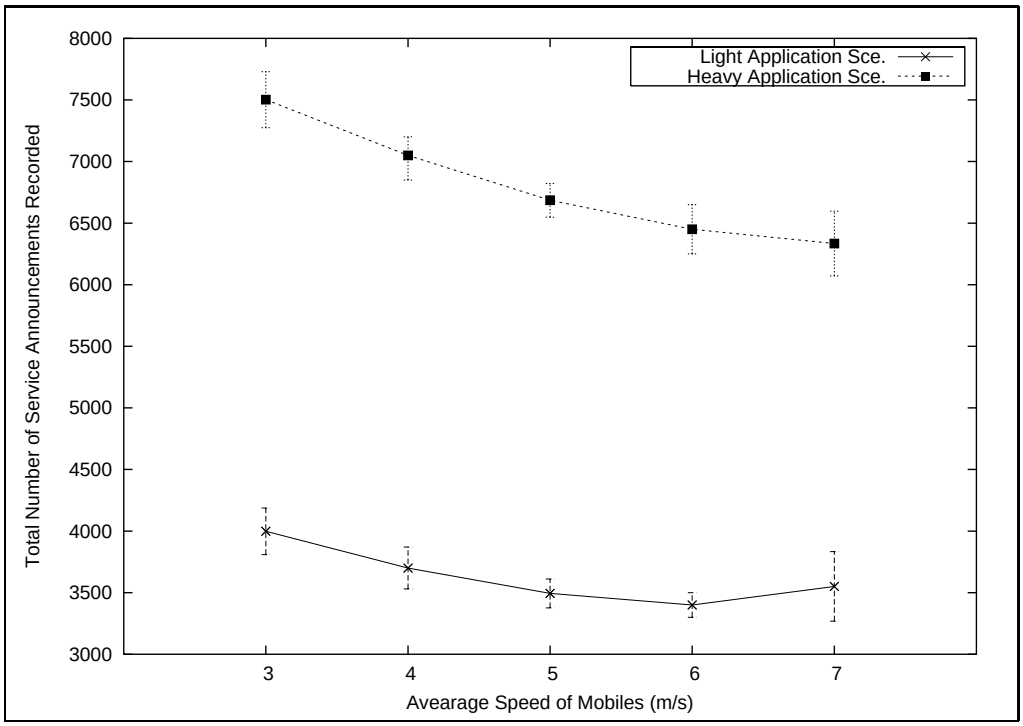


Figure 6.10. Total number of service instances recorded as average mobile speed changes

away from each other.

Results are presented in Table 6.5 for the classical *Packet Delivery Ratio* (PDR) metric of all 10 runs. The average is also given. It is obvious that DSR has much better packet delivery capability in this configuration. There are a couple of reasons for this result.

First and major one is because of the simplistic approach of routing defined in SeMA (See Section 4.4.3). SeMA routing agent does not take further corrective actions to remember the route loss of packets and tries whatever healing it has to offer for each and every other packet on the stream. However, for the route loss, DSR tries a new route discovery to facilitate new packets with fresh and valid source routes. Currently, the only way to achieve this is to increase the number of service announcement repetitions, whose results are analyzed in Section 6.5.4. Another reason for the lost in packets in the scenario using SeMA is the switching to flooding. As routes start to be out of date, nodes start to try alternatives, as explained in Section 4.4.3.2,

Table 6.5. Packet delivery ratios resulted from DSR and SeMA routing

	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
SeMA	0.169	0.179	0.144	0.165	0.127	0.109	0.145	0.186	0.110	0.208
DSR	0.736	0.767	0.805	0.809	0.754	0.793	0.767	0.763	0.821	0.803
Average for SeMA	0.154									
Average for DSR	0.781									

(for one more time in the simulations), and switch to flooding if not successful. Since the 80 per cent of the nodes are involved in sessions, the flooding occurring sessions affect some of the normally routed session packets using that part of the network by creating contentions. In this sense, IEEE 802.11 broadcast traffic is more likely to affect other stations than unicast traffic. This is because broadcast traffic does not make use of RTS/CTS mechanism to prevent further collisions caused to hidden terminals.

7. CONCLUSIONS AND FUTURE WORK

In this thesis, a cross layer ad hoc network protocol is proposed to discover, bind to, and utilize services available on the network. Issues of service definition, service discovery, multihop routing and session management are addressed in this simple service aware protocol.

To evaluate the performance of the proposed architecture, a frequently used wireless network simulation software, GloMoSim is extended to include the designed algorithms. To construct an operating environment for the applications using this protocol, a realistic campus scenario with applications are designed and implemented to be used in simulations. These applications include printing, voice and video streaming services beside generic CBR and VBR applications.

Although offered protocol is not specifically intended to be designed to satisfy networking requests of classical IP stack users, the simulation environment and applications therein reflect such a scenario. This approach is chosen to show the generic applicability of the protocol in terms of service discovery and routing. Results show this applicability is possible with sacrificing some performance in packet routing. Using proposed protocol, applications are found to be able to discover and bind to valid services even in networks of a few hundred users. Use and tuning of the algorithms for some other alternative environments are investigated. A sensor network architecture backbone, for example, is offered in [82] making use of this proposed protocol in providing access to environmental monitoring services.

Using simulation results as the proof of concepts, some mechanisms and protocol parameters are to be revisited, and augmented if possible. Routing, for example, needs some extra care to better perform with route loss characteristics in mind. The routing performance may be enhanced to include mechanisms that will prevent frequent flooding of messages and make hosts remember corrective actions taken for recent route losses. This is important since routing performance has a significant effect on the overall

classical performance metrics like packet delivery ratio or throughput.

Efficient representation of XML instances for both host and service attributes is another issue to be carefully considered. Having well-compressed and compacted instances affects the transport performance of the protocol, since host and service instances are frequently used in protocol packets. This effect will increase its strength if more feature aware mechanisms will be introduced into the algorithms in the future.

Having a service and host attribute aware network layer offers unlimited capabilities for task specific networks. Because of time constraints, a feature aware network layer operation has not been implemented, but all the framework that will ease the job of anyone considering such an addition is provided. This addition may include popular approaches to ad hoc networking like *power efficient algorithms*, *QoS aware routing* or *location based services*.

The proposed protocol does not imply any direct compatibility for currently implemented TCP/IP stack using applications. The applications of ad hoc network that will use our protocol have to make use of the new primitives supplied to access and use the services. Access to an IP (or another) backbone network may be defined and offered as a service on the proposed network, since the routing itself is a service provided by network layers of protocol stacks. Such gateway services may well be provided by fixed hosts of ad hoc networks with a connection to the wired infrastructure.

APPENDIX A: SAMPLE HOST XML SCHEMA DOCUMENT

In this Appendix, a sample host XML schema document (XSD) is presented, which is used to validate a given host XML instance.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema targetNamespace="http://ics.yeditepe.edu.tr/tnl/2003/01/MANET"
 xmlns="http://ics.yeditepe.edu.tr/tnl/2003/01/MANET"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
 attributeFormDefault="unqualified" version="0.1">

   <xsd:element name="service" type="serviceType">

      <xsd:annotation>
        <xsd:documentation> The topmost element to define a service </xsd:documentation>
      </xsd:annotation>

   </xsd:element>

   <xsd:complexType name="serviceType">

      <xsd:annotation>
          <xsd:documentation>The type definition to define a service.</xsd:documentation>
      </xsd:annotation>

      <xsd:sequence>
        <xsd:element name="keyword" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                 <xsd:attribute name="attribute" type="xsd:string" use="required"/>
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>

      <xsd:attribute name="name" type="xsd:string" use="required"/>
    </xsd:complexType>

</xsd:schema>
```

APPENDIX B: SAMPLE APPLICATION CONFIGURATION FILE

In this Appendix, a sample application configuration file is presented. This file represents the light application scenario of the base problem that is used in the experiments of performance evalustion. Contents of the file are processed by the GloMoSim simulation software.

```
# LIGHT APPLICATION SCENARIO

SEMA OFFER  3 1MS 100S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                     <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 15 2MS 100S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                     <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 16 3MS 100S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                     <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 33 4MS 100S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                     <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 47 5MS 100S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                     <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|

SEMA OFFER  3 100001MS 200S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                      <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 15 100002MS 200S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                      <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 16 100003MS 200S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                      <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 33 100004MS 200S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                      <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 47 100005MS 200S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                      <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|

SEMA OFFER  3 203001MS 0S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                    <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 15 203002MS 0S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                    <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 16 200003MS 0S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                    <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 33 200004MS 0S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                    <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|
SEMA OFFER 47 200005MS 0S <s n="printer"><k a="location">Engineering</k><k a="papersize">A4</k>
                    <k a="postscript">yes</k><k a="maxResolution">600*600</k></s>|

SEMA ASK 6  188S  <s n="printer"><k a="filename">rfc.txt</k><k a="papersize">A4</k>
```

```
                                <k a="postscript">yes</k></s>|
SEMA ASK 8    2S  <s n="printer"><k a="filename">user-manual.pdf</k>
                                <k a="papersize">A4</k><k a="postscript">yes</k></s>|
SEMA ASK 9  138S  <s n="printer"><k a="filename">rfc.txt</k>
                                <k a="papersize">A4</k><k a="postscript">yes</k></s>|
SEMA ASK 11 108S  <s n="printer"><k a="filename">rfc.txt</k>
                                <k a="papersize">A4</k><k a="postscript">yes</k></s>|
SEMA ASK 13 294S  <s n="printer"><k a="filename">rfc.txt</k>
                                <k a="papersize">A4</k><k a="postscript">yes</k></s>|
SEMA ASK 20  57S  <s n="printer"><k a="filename">rfc.txt</k>
                                <k a="papersize">A4</k><k a="postscript">yes</k></s>|
SEMA ASK 30  32S  <s n="printer"><k a="filename">rfc.txt</k>
                                <k a="papersize">A4</k><k a="postscript">yes</k></s>|
SEMA ASK 39  81S  <s n="printer"><k a="filename">user-manual.pdf</k>
                                <k a="papersize">A4</k><k a="postscript">yes</k></s>|
SEMA ASK 37 190S  <s n="printer"><k a="filename">user-manual.pdf</k>
                                <k a="papersize">A4</k><k a="postscript">yes</k></s>|
SEMA ASK 45 154S  <s n="printer"><k a="filename">book-small.pdf</k>
                                <k a="papersize">A4</k><k a="postscript">yes</k></s>|

SEMA OFFER 1 6MS 100S       <s n="VoSeMA"><k a="calee">1</k></s>|
SEMA OFFER 1 100006MS 200S  <s n="VoSeMA"><k a="calee">1</k></s>|
SEMA OFFER 1 200006MS 0S    <s n="VoSeMA"><k a="calee">1</k></s>|
SEMA ASK 3 40S              <s n="VoSeMA"><k a="codec">G.723.1</k>
                               <k a="duration">20S</k><k a="calee">1</k></s>|

SEMA OFFER 19 7MS 100S          <s n="VoSeMA"><k a="calee">19</k></s>|
SEMA OFFER 19 100007MS 200S     <s n="VoSeMA"><k a="calee">19</k></s>|
SEMA OFFER 19 200007MS 0S       <s n="VoSeMA"><k a="calee">19</k></s>|
SEMA ASK 44 110S                <s n="VoSeMA"><k a="codec">G.723.1</k><k a="duration">20S</k>
                                   <k a="calee">19</k></s>|

SEMA OFFER 49 8MS 100S        <s n="VoSeMA"><k a="calee">49</k></s>|
SEMA OFFER 49 100008MS 200S   <s n="VoSeMA"><k a="calee">49</k></s>|
SEMA OFFER 49 200008MS 0S     <s n="VoSeMA"><k a="calee">49</k></s>|
SEMA ASK 17 250S              <s n="VoSeMA"><k a="codec">G.723.1</k><k a="duration">20S</k>
                                   <k a="calee">49</k></s>|

SEMA OFFER 31 9MS 100S       <s n="videoStream"><k a="target">31</k><k a="encoder">MPEG-4</k></s>|
SEMA OFFER 31 100009MS 200S  <s n="videoStream"><k a="target">31</k><k a="encoder">MPEG-4</k></s>|
SEMA OFFER 31 200009MS 0S    <s n="videoStream"><k a="target">31</k><k a="encoder">MPEG-4</k></s>|
SEMA ASK 27 15S              <s n="videoStream"><k a="duration">60S</k><k a="media">StarWarsIV-lowres</k>
                               <k a="target">31</k><k a="encoder">MPEG-4</k></s>|
```

REFERENCES

1. Akyıldız, I. F., W. Su, Y. Sankarasubramaniam and E. Çayırcı, "Wireless Sensor Networks: A Survey", *Computer Networks (Elsevier)*, Vol. 38, No. 4, pp. 393–422, March 2002.

2. Weiser, M., "Some Computer Science Issues in Ubiquitous Computing", *Communications of the ACM*, Vol. 36, No. 7, pp. 75–85, July 1993.

3. Satyanarayanan, M., "Pervasive Computing: Vision and Challenges", *IEEE Personal Communications*, pp. 10–17, August 2001.

4. Tanenbaum, A. S., *Computer Networks*, Prentice-Hall Inc., New Jersey, USA, 1996.

5. *IETF Mobile Ad Hoc Networks (MANET) Official Charter*, `http://www.ietf.org/html.charters/manet-charter.html (Last Checked on February 26`th`, 2003)`.

6. Goldsmith, A. J. and S. B. Wicker, "Design Challenges for Energy-Constrained Ad Hoc Wireless Networks", *IEEE Wireless Communications*, Vol. 9, No. 4, pp. 8–25, August 2002.

7. *Longman Dictionary of Contemporary English (Second Edition)*, Longman Group UK Limited, Essex, England, 1992.

8. Perkins, C. E., *Ad Hoc Networking*, Addison Wesley Professional, Boston, USA, 2000.

9. Perkins, C., *Mobile Ad Hoc Networking Terminology*, Internet draft (work in progress), IETF, 1998, `http://www.iprg.nokia.com/~charliep/txt/manet/term.txt (Last Checked on February 26`th`, 2003)`.

10. Rappaport, T. S., *Wireless Communications: Principles and Practice (Second Edi-*

tion), Prentice-Hall Inc., New Jersey, USA, 2001.

11. Kleinrock, L. and F. A. Tobagi, "Packet Switching in Radio Channels: Part I – Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics", *IEEE Transactions on Communications*, Vol. 23, No. 12, pp. 1400–1416, December 1975.

12. Haas, Z. J., J. Deng, B. Liang, P. Papadimitratos and S. Sajama, "Wireless Ad Hoc Networks", J. Proakis (Editor), *Encyclopedia of Telecommunications*, John Wiley and Sons Inc., 2002.

13. Karn, P., "MACA: A New Channel Access Method for Packet Radio", *Proceedings of ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pp. 134–140, New York, USA, April 1990.

14. Bharghavan, V., A. Demers, S. Shenker and L. Zhang, "MACAW: A Media Access Protocol for Wireless LANs", *Proceedings of the SIGCOMM'94*, pp. 212–225, London, UK, August 1994.

15. Fullmer, C. L. and J. J. Garcia-Luna-Aceves, "Floor Acquisition Multiple Access (FAMA) for Packet-Radio Networks", *Proceedings of the SIGCOMM'95*, pp. 262–273, Cambridge, USA, October 1995.

16. *Radio Equipment and Systems (RES); High Performance Radio Local Area Network (HIPERLAN) Type 1, Functional Specification*, Standard Document ETS 300 652, European Telecommunications Standards Institute, Sophia Antipolis, France, October 1996.

17. *IEEE/IEC Std 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Standard document, The Institute of Electrical and Electronics Engineers, New York, USA, August 1999.

18. *Broadband Radio Access Networks (BRAN), HIPERLAN Type 2, System*

Overview, Standard Document TR 101 683, European Telecommunications Standards Institute, Sophia Antipolis, France, August 2000.

19. Negus, K. J., J. Waters, J. Tourrilhes, C. Romans, J. Lansford and S. Hui, "HomeRF and SWAP: Wireless Networking for the Connected Home", *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 2, No. 4, pp. 28–37, October 1998.

20. *Bluetooth Special Interest Group: Specification of the Bluetooth System, Version 1.1*, February 2001, `http://www.bluetooth.com (Last Checked on February 26`th`, 2003)`.

21. Liu, J., D. M. Nicol, L. F. Perrone and M. Liljenstam, "Towards High Performance Modeling of the 802.11 Wireless Protocol", *Proceedings of the 33rd Winter Simulation Conference WSC'01*, pp. 1315–1320, Arlington, USA, December 2001.

22. Camp, T., J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research", *Wireless Communications & Mobile Computing (WCMC), Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, Vol. 2, No. 5, pp. 483–502, 2002.

23. Johnson, D. B. and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", T. Imielinski and H. Korth (Editors), *Mobile Computing*, Vol. 353, chap. 5, pp. 153–181, Kluwer Academic Publishers, 1996.

24. Royer, E. M., P. M. Melliar-Smith and L. E. Moser, "An Analysis of the Optimum Node Density for Ad Hoc Mobile Networks", *Proceedings of the IEEE International Conference on Communications ICC'01*, Vol. 3, pp. 857–861, Helsinki, Finland, June 2001.

25. Haas, Z. J., "A New Routing Protocol for Reconfigurable Wireless Networks", *Proceedings of the IEEE International Conference on Universal Personal Communications ICUPC'97*, pp. 562–565, San Diego, USA, October 1997.

26. Bettstetter, C., "Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks", *Proceedings of ACM International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems MSWiM'01*, pp. 19–27, Rome, Italy, July 2001.

27. Liang, B. and Z. J. Haas, "Predictive Distance-Based Mobility Management for PCS Networks", *Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications INFOCOM'99*, pp. 1377–1384, New York, USA, March 1999.

28. Tuğcu, T. and C. Ersoy, "How A New Realistic Mobility Model Can Effect the Relative Performance of a Mobile Networking Scheme", To appear in Wireless Communications & Mobile Computing Journal, Wiley Publishers.

29. Xie, H., S. Tabbane and D. Goodman, "Dynamic Location Area Management and Performance Analysis", *Proceedings of 43rd IEEE Vehicular Technology Conference VTC'93*, pp. 536–539, Secaucus, USA, May 1993.

30. Lopez, M. S., "Mobility Models Page", `http://www.disca.upv.es/misan/mobmodel.htm (Last Checked on February 26`th`, 2003)`.

31. Hong, X., M. Gerla, G. Pei and C. C. Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks", *Proceedings of ACM/IEEE International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems MSWiM'99*, pp. 53–60, Seattle, USA, August 1999.

32. Penttinen, A., "Research On Ad Hoc Networking: Current Activity And Future Directions", `http://citeseer.nj.nec.com/533517.html (Last Checked on February 26`th`, 2003)`.

33. Royer, E. M. and C. K. Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", *IEEE Personal Communications*, pp. 46–55, April 1999.

34. Bertsekas, D. and R. Gallager, *Data Networks (Second Edition)*, Prentice-Hall Inc., New Jersey, USA, 1992.

35. Cheng, C., R. Riley, S. P. R. Kumar and J. J. Garcia-Luna-Aceves, "A Loop-Free Bellman-Ford Routing Protocol Without Bouncing Effect", *Proceedings of ACM SIGCOMM'89*, pp. 224–237, Austin, USA, September 1989.

36. Garcia-Luna-Aceves, J. J., "Loop-Free Routing Using Diffusing Computations", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 1, pp. 130–141, February 1993.

37. Perkins, C. and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", *Proceedings of ACM SIGCOMM'94*, pp. 234–244, London, UK, August 1994.

38. Chen, T. W. and M. Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks", *Proceedings of the IEEE International Conference on Communications ICC'98*, pp. 171–175, Atlanta, USA, June 1998.

39. Pei, G., M. Gerla and T.-W. Chen, "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks", *Proceedings of the IEEE International Conference on Communications ICC'00*, pp. 70–74, New Orleans, USA, June 2000.

40. Iwata, A., C. C. Chiang, G. Pei, M. Gerla and T.-W. Chen, "Scalable Routing Strategies for Ad Hoc Wireless Networks", *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, Vol. 17, No. 8, pp. 1369–1379, August 1999.

41. Chiang, C. C., H. K. Wu, W. Liu and M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel", *Proceedings of IEEE Singapore International Conference on Networks SICON'97*, pp. 197–211, Singapore, April 1997.

42. Clausen, T., P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum and L. Viennot, *Optimized Link State Routing Protocol*, Internet draft (work in progress), IETF, December 2002, `http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-07.txt (Last Checked on February 26th, 2003)`.

43. Joa-Ng, M. and I. T. Lu, "A Peer-to-Peer Zone-Based Two-Level Link State Routing for Mobile Ad Hoc Networks", *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, Vol. 17, No. 8, pp. 1415–1425, August 1999.

44. Sinha, P., R. Sivakumar and V. Bharghavan, "CEDAR: A Core-Extraction Distributed Ad Hoc Routing Algorithm", *Proceedings of IEEE Conference on Computer Communications INFOCOM'99*, pp. 202–209, New York, USA, March 1999.

45. Murthy, S. and J. J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks", *ACM Mobile Networks and Applications Journal, Special Issue on Routing in Mobile Communication Networks*, Vol. 1, No. 2, pp. 183–197, October 1996.

46. Perkins, C. E., E. M. Belding-Royer and S. R. Das, *Ad Hoc On-Demand Distance Vector (AODV) Routing*, Internet draft (work in progress), IETF, February 2003, `http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt (Last Checked on February 26th, 2003)`.

47. Johnson, D. B., D. A. Maltz, Y. C. Hu and J. G. Jetcheva, *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, Internet draft (work in progress), IETF, February 2002, `http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt (Last Checked on February 26th, 2003)`.

48. Park, V. D. and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", *Proceedings of IEEE Conference on Com-*

puter Communications INFOCOM'97, pp. 1405–1413, Kobe, Japan, April 1997.

49. Toh, C. K., "Associativity-Based Routing For Ad Hoc Mobile Networks", *Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems, Kluwer Academic Publishers*, Vol. 4, No. 2, pp. 103–139, March 1997.

50. Dube, R., C. Rais, K. Wang and S. Tripathi, "Signal Stability Based Adaptive Routing for Ad-Hoc Mobile Networks", *IEEE Personal Communications*, Vol. 4, No. 1, pp. 36–45, February 1997.

51. Ko, Y.-B. and N. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks", *Proceedings of the Fourth ACM/IEEE International Conference on Mobile Computing and Networking MobiCom'98*, pp. 66–75, Dallas, USA, October 1998.

52. Haas, Z. J., J. Y. Halpern and L. Li, "Gossip-Based Ad Hoc Routing", *Proceedings of IEEE Conference on Computer Communications INFOCOM'02*, Vol. 3, pp. 1707–1716, New York, USA, June 2002.

53. Haas, Z. J., M. R. Pearlman and P. Samar, *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*, Internet draft (work in progress), IETF, July 2002, `http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-zrp-04.txt` (`Last Checked on February 26`th`, 2003`).

54. Nikaein, N., H. Labiod and C. Bonnet, "DDR- Distributed Dynamic Routing Algorithm for Mobile Ad Hoc Networks", *Proceedings of the First Annual Workshop on Mobile Ad Hoc Networking & Computing MobiHOC'00*, pp. 19–27, Boston, USA, August 2000.

55. Guttman, E., C. Perkins, J. Veizades and M. Day, *Service Location Protocol, Version 2*, RFC 2608, IETF, 1999.

56. Edwards, W. K., *Core Jini (Second Edition)*, Prentice-Hall Inc., New Jersey, USA, 2000.

57. *Sun Microsystems Inc.*, `http://www.sun.com (Last Checked on February 26th, 2003)`.

58. *Salutation Consortium*, `http://www.salutation.org (Last Checked on February 26th, 2003)`.

59. *Universal Plug and Play Forum*, `http://www.upnp.org (Last Checked on February 26th, 2003)`.

60. Eustice, K. F., T. J. Lehman, A. Morales, M. C. Munson, S. Edlund and M. Guillen, "A Universal Information Appliance", *IBM Systems Journal*, Vol. 38, No. 4, pp. 575–602, 1999.

61. *Bluetooth Specification Part E. Service Discovery Protocol (SDP)*, November 1999, `http://www.bluetooth.com (Last Checked on February 26th, 2003)`.

62. Czerwinski, S., B. Y. Zhao, T. Hodes, A. Joseph and R. Katz, "An Architecture for a Secure Service Discovery Service", *Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking MobiCOM'99*, pp. 24–35, Seattle, USA, August 1999.

63. Adjie-Winoto, W., E. Schwartz, H. Balakrishnan and J. Lilley, "The Design and Implementation of an Intentional Naming System", *Symposium on Operating Systems Principles*, pp. 186–201, Kiawah Island, USA, December 1999.

64. Baydere, Ş. and M. A. Ergin, "An Architecture for Service Access in Mobile Ad Hoc Networks", *Proceeding of the IASTED Wireless and Optical Communications*, pp. 392–397, Banff, Canada, July 2002.

65. Baydere, Ş. and M. A. Ergin, "A Model for Dynamic Service Discovery in Wireless Ad Hoc Networks", *Proceeding of the Sixth Symposium on Computer Networks*,

pp. 120–128, Gazimagusa, KKTC, June 2001.

66. Bray, T., J. Paoli, C. M. Sperberg-McQueen and E. Maler, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C recommendation, W3C, October 2000, `http://www.w3.org/TR/2000/REC-xml-20001006 (Last Checked on February 26th, 2003)`.

67. Fallside, D. C., *XML Schema Part 0: Primer*, W3C recommendation, W3C, May 2001, `http://www.w3.org/TR/2001/REC-xmlschema-0-20010502 (Last Checked on February 26th, 2003)`.

68. Bajaj, L., M. Takai, R. Ahuja, R. Bagrodia and M. Gerla, *GloMoSim: A Scalable Network Simulation Environment*, Tech. Rep. 990027, UCLA CSD, May 1999.

69. Bagrodia, R., R. Meyer, M. Takai, Y. an Chen, X. Zeng, J. Martin and H. Y. Song, "PARSEC: A Parallel Simulation Environment for Complex Systems", *IEEE Computer*, Vol. 31, No. 10, pp. 77–85, October 1998.

70. *QualNet from Scalable Network Technologies Inc.*, `http://www.qualnet.com (Last Checked on February 26th, 2003)`.

71. *OPNET Modeler from OPNET Technologies Inc.*, `http://www.opnet.com (Last Checked on February 26th, 2003)`.

72. *The Network Simulator (ns-2)*, `http://www.isi.edu/nsnam/ns (Last Checked on February 26th, 2003)`.

73. Deutsch, P., *DEFLATE Compressed Data Format Specification, Version 1.3*, RFC 1951, IETF, May 1996.

74. Liefke, H. and D. Suciu, *XMill: An Efficient Compressor for XML Data*, Tech. Rep. MS-CIS-99-26, University of Pennsylvania, 1999.

75. Sundaresan, N. and R. Moussa, "Algorithms and Programming Models for Efficient

Representation of XML for Internet Applications", *Computer Networks (Elsevier)*, Vol. 39, No. 5, pp. 681–697, August 2002.

76. *The Expat XML Parser*, `http://expat.sourceforge.net (Last Checked on February 26`th`, 2003)`.

77. Schulz, T., *Voice Over IP*, White paper, Eicon Technology Corporation, February 2000, `http://www.eicon.com/disvVoIPpri/whtpap4.htm (Last Checked on February 26`th`, 2003)`.

78. *Voice Over IP - Per Call Bandwidth Consumption*, Tech Note 7934, Cisco Systems Inc., December 2002.

79. Fitzek, F. H. and M. Reisslein, *MPEG-4 and H.263 Video Traces for Network Performance Evaluation*, Tech. Rep. TKN-00-06, Technical University of Berlin, Telecommunication Network Group, October 2000.

80. Jain, R., *The Art of Computer Systems Performance Analysis*, John Wiley and Sons Inc., New York, USA, 1991.

81. Das, S. R., C. E. Perkins and E. E. Royer, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks", *Proceedings of IEEE Conference on Computer Communications INFOCOM'00*, pp. 3–12, Tel Aviv, Israel, March 2000.

82. Baydere, Ş., M. A. Ergin and Ö. Durmaz, "Query Driven Sensor Networks for Environmental Monitoring Applications", Submitted to the Computer Networks (Elsevier), Special Issue on Wireless Sensor Networks.