WiFi Ad-hoc Message Propagation over GPRS Networks

M.Sc. in Networking & Internet Systems
September 2006

By

Yehia El khatib, B.Sc.

Additional Material:
http://www.lancs.ac.uk/postgrad/elkhatib/prj/index.htm

Disclaimer

I certify that the material contained in this dissertation is my own work and does not contain significant portions of unreferenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project and all associated documentation.

In the case of electronically submitted work, I also consent to this work being stored electronically and copied for assessment purposes, including the department's use of plagiarism detection systems in order to check the integrity of assessed work.

Date: _____

Signed: _____

Table of Contents

Table of Contents

Table of Contents

Abstract

Mobile ad hoc networks (MANETs) are networks that can be created spontaneously with no prior set up. Arbitrary nodes can form such networks by intercommunicating directly in a peer-to-peer fashion using wireless communication technologies, such as IEEE 802.11 (or WiFi).

The number of WiFi-enabled handheld devices is escalating as the services they offer attract an increasing number of customers. These devices have the capability of creating MANETs regardless of any infrastructure. Such networks can be used to circulate data across the network at high rates.

However, MANETs are networks of very special nature. They feature delays of unexpected lengths, high retransmission rates (due to radio interference, obstacles, and packet collisions), and unpredictable node relocation. For these reasons, MANETs tend to have unstable topologies.

In this project, we present a solution for exchanging messages in MANETs formed by WiFi-enabled handheld devices. We implement the proposed solution using the network simulator ns2, and investigate its behaviour and performance under different network conditions.

Acknowledgements

First and foremost, I thank God, the most Merciful. It is only by His blessings that I am able to accomplish anything.

I would like to express my sincere gratitude to Prof. Stephen Pink for his continuous encouragement. He has believed in me right from the beginning and that has meant a lot to me.

I would also like to acknowledge the unwavering support of Dr. Christopher Edwards. I have been very fortunate to have Dr. Edwards as a supervisor following Dr. Pink's unexpected illness. I thank Dr. Edwards for his insightful and precious comments. I am deeply grateful for his guidance and understanding throughout the course of the project.

I thank Benjamin Bappu at British Telecom for his patience and support. I thank Matthew Jakeman for his help with UNIX and ns2. I also thank George Metaxas for sharing his knowledge and for his huge assistance with ns2 and C++.

I would like to express my heartfelt gratitude towards Prof. M. Nazeeh El Derini. I still recall bits and pieces of his inspiring lectures back when I was an undergraduate at Alexandria University. I owe a great deal of my love to computer networks and computer science in general to him, and to the rest of the faculty at Alexandria University.

I wish to thank my family. I will always be indebted to my parents Saíd and Asmaa, my brother Mohamed, and my two sisters Mona and Radwa for their unconditional love and endless support.

Last, but certainly not least, I thank my wife-to-be Mariam for her love, support and encouragement. To her I dedicate this work.

List of Figures

List of Figures

List of Tables

1 Introduction

A mobile ad hoc network is a network of mobile nodes which is set up spontaneously with no prior infrastructure. The comprising nodes communicate directly with each other, and are collectively responsible for the set up, management, and maintenance of the network. Such networks are of a unique nature, and are becoming increasingly popular.

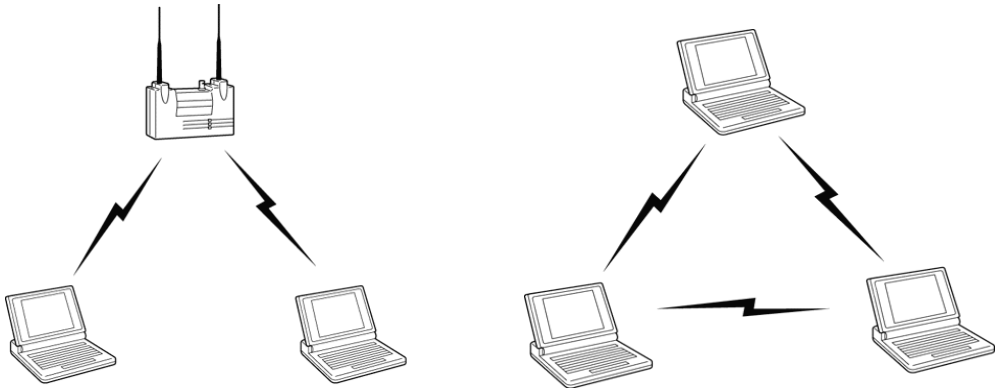
These networks tend to have very dynamic multi-hop topologies. There is no control or monitoring by central routers or servers, but each node acts as a router. Such properties qualify mobile ad hoc networks to extend the reach of networks beyond the geographical location of any infrastructure, and thus they facilitate a large variety of applications.

1.1 Background

Wireless connectivity has always been appealing to the experts and the public alike. It is easy to set up, convenient to use, and relatively economical. The IEEE 802.11 communication standards provide more reliability, higher bandwidth, and wider nominal range than other wireless communication standards such as Bluetooth [17]. The 802.11 standard is commonly known as Wi-Fi or WiFi (Wireless Fidelity).

The IEEE 802.11 standard specifies two different communication modes for wireless local area networks (LANs) [1]. Using the first mode, the wireless LAN consists of a set of nodes (wireless stations) which share a single coordination function. The coordination function is carried out by an access point (AP), which is a station that relays frames to and from the client nodes in its range. Such network created by the access point and the set of nodes it serves is usually referred to as an *infrastructure wireless LAN*. The access point may be used solely to interconnect a set of nodes. However, an access point is typically connected to other networks such as the Internet, providing a bridge between the set of client nodes and other networks.

The second mode consists of a set of nodes which can intercommunicate directly with each other. There is no need for an access point to establish a connection between the nodes. This mode is referred to as *ad hoc mode*, and a network based on this mode is called a Mobile Ad hoc Network (MANET). Figure 1.1 illustrates these modes of WiFi.



Infrastructure mode Ad hoc mode

Figure 1.1: The Two Modes of WiFi

(images from http://support.dell.com/support/edocs/network/079nk/specs.htm)

The designation “mobile” indicates that the nodes are free to move. The connectivity of any node with other nodes is prone to change as it moves: link quality changes, neighbouring nodes become out of range, and new nodes are found. The connectivity with other nodes could change even when the node is motionless. This is due to unpredictable changes in wireless link quality, and to the motion of other nodes in the same network. In that sense, nodes can join and leave a MANET without prior notice. For these reasons, a MANET is not just infrastructure-less but its spontaneous topology is also subject to unpredictable change.

The lack of any infrastructure or administration necessitates a MANET to be autonomic. A MANET has to be self-organizing, self-managing, and self-repairing. The nodes of a MANET are, thus, collectively involved in administration as much as they are in communication. Each node acts as a router, exchanging routing control messages (such as advertisements) and maintaining local data structures of network information.

In spite of MANETs’ instable nature, they offer distinctive advantages over conventional networks and over infrastructure wireless networks. Their spontaneity is a great value which makes them usable in applications where a network needs to be created “right now, right here” given only the availability of the participating nodes. The autonomic property of MANETs makes them adaptive and fault-tolerant: there is no single point of failure. Such networks can be very useful in commercial, military, intelligence collection, and personal applications.

1.2 Motivation

Wireless LANs are now widely used in houses, corporations and public venues. Wireless adaptors and access points are now widely availble at low prices. The main motivation behind the project is to develop an application which exploits this potential to provide exchange of content over spontaneous, autonomous wireless networks.

There are a number of wireless solutions that are available to use, such as Bluetooth, WiFi, WiMax, as well as some proprietary technologies. Out of these, WiFi stands out for several reasons. Excluding WiMax, WiFi offers the highest bandwidth levels. It also offers the widest communication range.

WiMax is the name given to the IEEE 802.16 communication standard. This technology provides wireless communication for MANs (metropolitan area networks) [11]. WiMax is much superior to WiFi in terms of data transmission rates and range. WiMax utilises multiple channels for communication, offering data rates of upto 350 Mbps. The WiMax communication range can reach to 50 kilometers due to the use of high power rates [37].

Notwithstanding all the capabilities of WiMax, it cannot be used in this project for many reasons. First and most importantly, WiMax is an infrastructure-based technology. It relies on very high power base stations. Providing such infrastructure is expensive and imposes geographical restrictions on the network. WiFi, on the other hand, facilitates building networks without any prior planning. Moreover, WiMax is not widely deployed and supported as WiFi.

WiFi is a well-established technology. WiFi infrastructure, in the form of "hot zones" served by access points, is widely available and would only be used for the function of delivering content from its initiator to the public. The content is then propagated over ad hoc networks made up of handheld devices. The application proposed by this project can actually be deployed in a context where there is no need for any infrastructure at all. More importantly, there are almost no interoperability problems with WiFi products of different vendors. These products are tested and certified by the WiFi Alliance♦ [41].

Providers are racing to attract potential WiFi customers by constantly installing wireless access points in public places such as coffee shops, hotels, airports, railway stations, corporate buildings, etc. Urban WiFi coverage is rapidly increasing in many cities around the world, particularly in the UK [68]. Several cities already have a city-wide"WiFi blanket", such as San Francisco [74], New York City [75], Houston [77], Philadelphia [72], Portland [79], Orlando [73], Louisiana [70], and Taipei [76]. Lancashire's Preston was the first city in the UK to offer city-wide WiFi connectivity in 2004 [67].

Advancements in mobile technology and in Lithium-ion batteries have facilitated supplying mobile phones with WiFi adaptors. Encouraged by the huge potential that the WiFi market has gained from the expanding coverage, almost all major mobile phone manufacturers have recently launched phones with WiFi capabilities. Currently, there are 29 different WiFi-enabled mobile phones available in the market (listed in Appendix E). Providers already have plans to exploit this new market. For example, Net2Phone and BridgePort Networks have collaborated in a project to provide Voice over IP (VoIP) to WiFi-enabled mobile phones [78].

The truth is that the number of handheld WiFi-enabled devices is increasing, and that there is a wireless access point around each corner. British Telecom, the sponsor of this project, is interested in providing a service that can exploit such potential. Their aim is to use these handheld devices to disperse commercial content. By using

♦ The WiFi Alliance is a non-profit organisation concerned with the interoperability of WiFi devices. The WiFi Alliance is also involved with promoting the WiFi technology and its products. Members of the organisation represent leading manufacturers of WiFi devices and software providers.

MANETs, providers such as BT can increase the number of mobile users who receive the content without investing in more infrastructure.

From an academic point of view, the project presents a solution that can be used to distribute discrete, time-tolerant content which demands high bandwidth over spontaneous, autonomic networks. The same concepts and decisions presented in this project can be used and applied in similar applications, such as sensor networks, disaster recovery, internal corporate multicasting and general broadcasting.

1.3 Problem Description

The importance of this project stems from the fact that MANETs are still under extensive research. There is no standard routing protocol, nor a standardised addressing scheme. A large number of reports emerge every year investigating different aspects of MANETs.

As this field is still growing, it is not easy to deploy an application before carefully studying its demands and how they can be met in a MANET. To achieve this, it was crucial to first define the application requirements as comprehensively as possible. Then, many of the properties of MANETs had to be examined, such as their unstable topologies, distributed administration, as well as link quality and interference. The details of this research and the decisions made in light of it are discussed in Chapter 2.

1.4 Objectives

Thorough research in the area of MANETs has been ongoing for a few years now. This research has kept spawning interesting, yet, unanswered questions. This is not out of the ordinary for an academic research, but in this case the research has not yet reached a point where protocols have been recognised as standards.

The overall goal of this project is to implement an application for MANETs, for which no standards yet exist. In order to achieve this goal, the following objectives have to be met:

- Defining an addressing scheme and a routing protocol;
- Developing a simulation of the network;
- Designing and developing an implementation of the application; and
- Evaluating the performance of the application in the simulated environment.

These objectives are discussed further in section 2.1.

1.5 Document Overview

This chapter sets the stage for the work that has been done in this project. It introduces MANETs and their research field. It also comments on the motivation behind the project, as well as the main goal and objectives. The rest of the document outlines the work that has been carried out. Chapter 2 illustrates the demands of the target application. The chapter then provides background information of MANETs and focuses on the areas which are most relevant to implementing the application. Chapter 3 discusses the simulation, and the design of the application. It then discusses

the implementation of the application in more detail. Chapter 4 lists and analyses the results obtained from evaluating the implemented application. Finally, Chapter 5 provides conclusions drawn from the project, and outlines areas of future work.

2 Problem Analysis & Literature Overview

This chapter provides an overall analysis of the target problem domain. First, an analysis of the requirements and goals of the application is presented. Then, the chapter discusses the two main decisions made in the project. For each of these decisions, we give an overview of the existing technologies for MANETs. We analyze these technologies and pick the one which best meets the requirements of the target problem domain.

2.1 Problem Definition

This section describes the target implementation. This description builds a basic understanding of the characteristics of the network into which we intend to deploy our implementation.

2.1.1 A Conceptual View

A large number of mobile nodes make up the target network. Each of these nodes possesses a set of data chunks we will refer to as messages. The nodes are concerned with exchanging these messages with each other so as to spread them across as many nodes as possible. However, the message exchange should be carried out in such a manner so as to keep track of who propagated which message.

As mentioned, the nodes are mobile. These nodes form MANETs as they move into each others' transmission range. Each node must adapt to the changes in its surroundings (nodes and links).

The target application is one which can be implemented in many fashions. Such implementations can not be real-time. On the contrary, they have to be quite time-tolerant. For example, the network can be used to propagate the latest news headlines or sports scores. Nevertheless, some constraints could be imposed to control the scope of the time and/or location reach of the messages. For example, the messages can have a limited hop count or lifetime after which they expire.

2.1.2 The Envisaged Scenario

We have chosen to focus on a particular instance implementation. This particular implementation is a commercial application where the messages are multimedia-rich advertisements. The nodes are WiFi-enabled mobile phones. Initial copies of the messages (adverts) are injected by the service provider into the crowds by forwarding them to a small set of the users. This initial step of dispatching the messages into the public is performed via the service provider's wireless access points.

The users are rewarded by the advertiser, the provider, or both. Such rewards can be in the form of free minutes offered by the provider, discounts and special offers from the advertiser, etc. The exact terms and conditions of the rewarding system are left up to the provider and the advertisers to agree upon; we aim at facilitating and keeping track of propagating the messages across the MANETs.

Figure 2.1 below, presented by BT, depicts the target scenario.

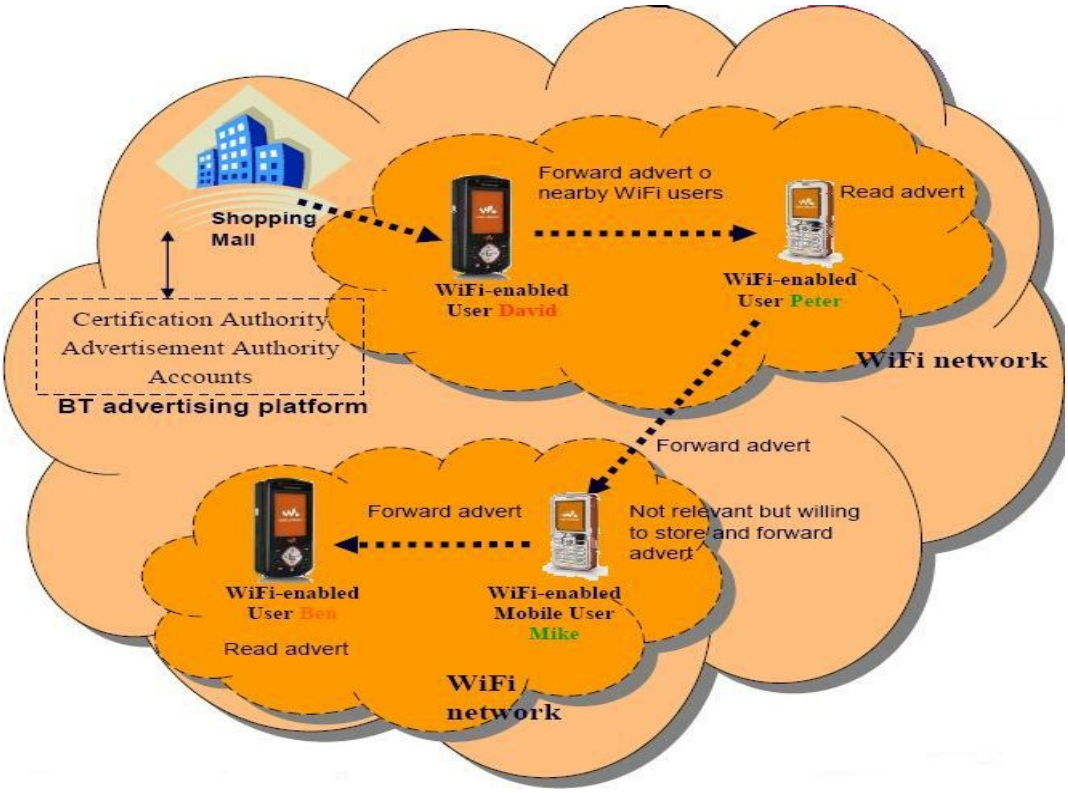


Figure 2.1: High Street Adverts Scenario

Following is a simple description of the scenario: User David is walking down the street. As he passes by a BT WiFi hot zone, the access point recognises his device as a customer device and sends him one or more adverts sponsored by some of the shops on the street. As he passes other users with WiFi-enabled devices, the adverts are forwarded to them, and so on. Users will be motivated to forward the messages by means of rewards as already mentioned.

2.1.3 Analysis of the Problem Domain

I. Message Initiation

For a mobile phone to receive the initial message from the provider's wireless access point, both devices have to be operating in the same mode. One way to achieve this is to have the customer's mobile phone switched to infrastructure mode for the entire duration of receiving the message from an access point, before returning to operate in ad hoc mode. As access points were originally designed to provide connectivity to users in infrastructure mode, the other option is to use access points that operate in both ad hoc and infrastructure modes.

The former option is not a very practical one since such a switch cannot be done through the application. This switch has to be done only when the user explicitly changes the operation mode of the wireless adaptor.

The latter option may also seem impractical. A wireless adaptor cannot operate in both modes at a given time because the 802.11 modes were not designed for such a purpose. The adaptor cannot switch between the two modes automatically through an application♦. This means that the wireless access points will be expected to contain a separate wireless adaptor which operates in ad hoc mode all the time. The feasibility of such a requirement might be a challenge, since the main objective of almost all access point set up by providers is to offer Internet coverage for WiFi-enabled devices. To provide such a service, the access points operate in infrastructure mode. Installing a separate wireless adaptor in currently deployed access points might indeed prove to be quite a challenge in face of the provider who wishes to implement this application. However, this issue is to be resolved by the provider and hence we shall not explore it further.

II. Naming and Addressing

The communication between the devices is anonymous, in the sense that there is no need for the sender or the receiver to know each other. The sender simply offers the message, and the receiver chooses to accept it or not without any knowledge of the sender. The receiver only cares for the genuinity of the offered message, and whether it is of interest to him/her or not. Hence, no naming service is required. The message is forwarded to any reachable device whose user is willing to accept it.

However, there is still a need to know the addresses of the communicating devices. Each device must obtain an IP address which does not conflict with the other surrounding devices. It is important to keep in mind that the surrounding devices are expected to be constantly changing. Such issues make addressing one of the main subjects inciting research into mobile ad hoc networks.

III. Routing

A further major obstacle is routing. A routing protocol is required to discover nearby devices, to update the routing data structures as devices join and leave the vicinity, and to find the best routes to optimise throughput, delay, and power consumption. This protocol must be robust against rapid topology changes. There are a vast number of routing protocols that have been designed for mobile ad hoc networks.

IV. Message Propagation Control

The last, but not least, of the problems to tackle is the control over message propagation. There are many concerns in the propagation of messages. The users will have profiles to describe the type of adverts they might be willing to accept. Advert metadata is used to classify the advert messages according to their details. The metadata will also contain the lifetime of the message among other things. The main concern here is to avoid duplicate messages. No user should recieve any

♦ For more information about operating in both modes, please refer to Appendix C.

particular advert more than once. Additionally, the mechanism should react gracefully to connections which fail due to topology changes.

2.1.4 Goals

The following goals are deduced from the above analysis.

- Establishing an addressing system which is light-weight and suitable for persistently changing wireless topologies;
- Determining which routing protocol best suits the target application; and
- Designing an application to control message propagation through session management and transaction tracing.

The first and the second goals will be discussed in the following two sections. The design of the application and the related implementation details will be delivered in the following chapter.

2.1.5 General Assumptions

We have developed certain assumptions that have helped us focus on accomplishing the aforementioned goals. Future work can stem from these assumptions, attempting to study them further and analyse their effect.

- The initial release of a message to the public through a wireless access point is not investigated further than what has been already mentioned. The feasibility of using access points which are capable of operating in both infrastructure and ad hoc modes at the same time can become a setback in the way of large scale implementation. Such study, however, is beyond the scope of this project. Therefore, we have assumed that messages are readily released to a group of users.
- Certain aspects of the application were not implemented, based on the assumption that they can be easily added on later. The project focuses on the lower layers of the application (addressing and routing) and part of the higher layer (responsible for governing the message propagation). Another project within the Computing Department of Lancaster University focuses on the higher layer issues of the application. This is mentioned in section 2.4.

  One example of such missing features is the cryptography mechanisms required for authentication (to avoid SPAM). Another example is the management of the user profiles, and how it is compared against incoming message metadata to decide whether to accept or reject the message. Clearly, the integration of such functionality into the application is necessary before deployment. However, in evaluating the application we will ignore such aspects and merely focus on the bare message exchanging elements of the application.

2.1.6 Methodology

The declared goals were achieved by following the next steps:

- Studying the IEEE Standard Specification for 802.11. Sufficient understanding of the WiFi standards was vital before embarking on decision making. Topics of interest were mostly related to the 802.11 MAC layer, such as device discovery, framing, and broadcasting.

- Investigating the demands of the target network. The main properties of the target scenario were outlined. These properties were the guidelines in making the decisions regarding addressing and routing.
- Reviewing the current addressing schemes. Addressing at different levels was investigated before settling on IP addressing. Next, a large number of reports on IP addressing in MANETs were studied since there is no standard scheme. The main strengths and weaknesses of each were highlighted and then used to select an addressing scheme which fits the network demands.
- Assessing the MANET routing protocols. Dozens of routing protocols have been developed for MANETs, only two of which are de facto standards (DSR and AODV). All these routing protocols were considered before selecting a small group of them as candidates. The protocols from this set are compared with each other in the context of the target network.
- Designing the application. The designed application is the mechanism responsible for exchanging messages between the users. Its main functions include avoiding duplicate messages, session management, and transaction tracing.
- Completing an evaluation of the application over the selected schemes. A simulation of the target network was generated to evaluate the behaviour and performance of the application using the chosen addressing and routing schemes.

2.2 Addressing

One of the crucial decisions to be made is on the matter of addressing of nodes, without which no communication is possible. In order to decide which address allocation scheme to use, we had to determine the heuristics of addressing in mobile ad hoc networks. This task consisted of studying the restrictions of addressing in the mobile ad hoc environment, and examining different proposals that have been suggested to solve this issue.

2.2.1 Link Layer Vs Network Layer Addressing

Physical addresses are sufficient for establishing connections between devices. They are also unique, as long as they abide by the IEEE MAC addressing standards. However, if physical addresses are used the deployment of the application on real devices would necessitate programming at the MAC layer. Programming at a level this low is usually very time-consuming. It can also be very inconvenient and impractical for anything other than small applications.

Hence, we have decided to use logical addressing. Typically, an IP address has two purposes: identification and routing. An IP address must act as a unique identifier of a host across a network. In infrastructure-based networks (wired and wireless), IP addresses are usually assigned from a pool of available IP addresses. This is done, by a DHCP server for example, in order to maintain a hierarchical addressing space. However, MANETs have no infrastructure, and usually no central authority to assign addresses. Thus, address space in a MANET is flat and IP addresses lack any associated routing information. In other words, an IP address in a MANET merely acts as an identifier of a particular device.

Network-layer addressing would enable easy programming and would provide the application with the flexibility of logical addressing. This, however, would require an automatic address configuration scheme suitable for such networks. Since topologies are quite dynamic, the scheme must not require too much bandwidth or time when nodes join or leave the network. The scheme is also necessary to deal swiftly with merging and partitioning networks. Further, the scheme would preferably be independent of the routing protocol used. This will not be challenging since the logical address in a MANET is merely an identifier and does not signify any routing information at all.

The required logical address can simply be in the form of integer tags, since MANETs are infrastructure-less networks which are disconnected from other networks (although this is not necessarily the case). However, IP addressing was chosen for the flexibility and extensibility it offers. All current mobile devices readily support IP.

2.2.2 The Desired Addressing Scheme

As it continues moving, a mobile node can move out of range of a certain MANET and into the range of another one. The IP address of the device may cause a conflict with another device in the other MANET. Moreover, a mobile node can at times be in the range of more than one network, merging them into one. In this case, the IP address it obtains must be unique within all networks it is a member of.

Considering the properties of MANETs, an addressing scheme which has the following characteristics is required:

- *Dynamic*: The scheme should allow each node to obtain an IP address dynamically without using any predefined IP addresses.
- *Unique*: The addresses obtained by the nodes should be unique in the network. This can be done either proactively (by keeping track of unused addresses) or reactively (by detecting duplicate addresses and resolving such conflicts).
- *Low overhead*: It is highly desirable for the scheme to use the least bandwidth possible, and to provide a node with a unique IP address in bound time.
- *Robust*: The high mobility of the nodes leads to a constant, rapid activity of joining and leaving, and to network merges and partitioning. The desired scheme should deal with such changes in topology with minimal disruption to already configured devices.

2.2.3 A Review of the Current Proposed Schemes

Less work has been done on MANET addressing than routing. Current addressing schemes can be divided into two main types, the first of which is static addressing where addresses are manually assigned to the devices (by an administrator). Most routing protocols assume this sort of addressing. This approach is not feasible for the cumbersome administration it involves, in addition to its poor utilisation of available address space. Moreover, one of the key strengths of MANETs is that they are non-administered spontaneous networks.

On the other hand, auto-configuration addressing schemes provide means for dynamic address assignment. Such schemes can be designed to provide more information about the devices (such as the network which the device is a member of, location-based information, etc.). Auto-configuration schemes can be either centralised or distributed. DHCP, for instance, is a centralised auto-configuration addressing scheme. However,

DHCP cannot be used in MANETs. Nodes in a MANET are typically mobile, and hence access to a fixed DHCP server is nearly impossible to ensure at any given time.

Different addressing schemes have been developed for the mobile ad hoc environment. Although there is an abundance of proposed addressing schemes, there is not a single standard auto-configuration scheme. The proposed schemes can be classified as centralised or distributed. The centralised ones elect a node or a group of nodes to be responsible for address space utilization, while distributed schemes rely on sharing this responsibility amongst all nodes. The schemes may also be classified in terms of how duplicate addresses are detected. All of the schemes suggested up until the last couple of years relied on reaction to duplicate address detection. Recently, there have been a few efforts to develop proactive duplicate address avoidance.

One of the first schemes proposed was Duplicate Address Detection (DAD) [29]. This is a reactive distributed scheme where a node picks an IP address at random or an IP address one which is derived from some readily available information such as the device's MAC address. The node then checks whether this IP address is already in use or not. The mechanism works as follows: after determining the initial IP address, the new node sends a message to its neighbours requesting a route to the selected IP address. The node then listens for an echo to its message for a defined period of time. If the timeout expires, the node can then claim the IP address as its own. If, however, a reply is received then the node has to pick another address and repeat the same procedure again.

It is worth noting that during this procedure, a new node takes on a temporary IP address from an address pool reserved particularly for this purpose. This pool is defined as part of the addressing scheme, and is only used for new nodes until they acquire a permanent address. Obviously, the larger this pool is the less address space usable for allocation.

Although this scheme might indeed induce very low overhead as it promises, there are a couple of concerns. First, wireless environments are generally unreliable. Thus, a timeout does not certainly indicate the lack of a reply, as it might indicate message exchange failure. Perkins et al. have suggested that the new node can repeat the procedure more than once before safely assuming that the address is free to be taken. Nevertheless, the timeout period is not an easy parameter to set. Unsurprisingly, delays between different nodes in the same MANET may vary to a great extent. Short timeouts may not give nodes the chance to reply indicating their acquisition of the address, while long timeouts will cause the scheme to be quite time-consuming which is not desirable. New nodes in particular have no idea about the characteristics of the network they join. In order to calculate a fairly good timeout period, a new node should consider the size of the network and the quality of the links (average round trip delay, interference, etc). Secondly, this addressing scheme has to be associated with a mechanism to resolve duplicate address conflicts when networks merge. This is discussed later in this section.

There are several addressing schemes very similar to DAD. One example is the name-based scheme suggested in [21], where addresses are acquired according to the hashed value of the host name.

Weak DAD has been proposed by Vaidya as a more realistic version of DAD [36]. This scheme allows more than one node to share a single IP address as long as correct routing is maintained. The scheme distinguishes different nodes on the basis of a key, which is a unique value set by the node. This key is used only in control messages, and hence introduces no added routing overhead. The key can be the device's MAC address, a large random key, or the node's public key if Public Key Infrastructure is implemented at the network layer. The scheme improves on DAD, but requires complex integration with the routing protocol employed.

Sun and Belding-Royer suggested a scheme similar to DAD, but which is a little more centralised [33]. Using their scheme, each MANET elects two nodes as the primary and backup address authorities. These nodes keep track of the address space utilisation in the MANET. They also help identify the network from other neighbouring networks by periodically broadcasting a network identifier. This helps in identifying network merges by simply recognising foreign network identifiers. As in DAD, a new node picks an address and then sends an Address Request (AREQ) message asking for the permission to use this address. If no Address Reply (AREP) message is received in a defined period of time, the node assumes that the address is free for use and registers with the address authority. Otherwise, the node picks another address and goes through the DAD process again.

Unlike reactive techniques, proactive schemes are very few. The key advantage of proactive techniques is that they are not timeout-based. This characteristic is quite suitable for MANETs as discussed before. One technique has been presented in [35]. The scheme is a centralised one where a leader node is elected in each MANET. This leader node keeps track of the addresses in use, and assigns new IP addresses accordingly. A joining node asks the closest node (referred to as the 'agent') for an address, and the agent forwards the request to the leader. The leader also acts as a beacon to identify the network. Merges and partitions are recognised in a manner similar to Sun et al's scheme.

Another protocol which is proactive in concept is Dynamic Registration and Configuration Protocol (DRCP) [23]. Each node acquires an address pool. New nodes get their own address pool by requesting half of the pool of one of its neighbors. The protocol is efficient and incurs a finite latency, yet it does not provide a mechanism to handle partitions and merges.

2.2.4 Conclusion

Initially, we have decided to employ the DAD scheme in obtaining an IP address for a newly joined node. DAD produces the least network overhead, and it fulfills all of the requirements mentioned in section 2.2.2. Apart from Weak DAD, all the other techniques can add a high network overhead in dynamic networks.

The DAD scheme, however, has another crucial shortcoming: it can not deal with merges and partitions. The DAD protocol is completely stateless. Without any modification, the DAD protocol would deal with an address conflict in a manner illustrated by the following example. Consider networks I and II shown below. Node A in network I and node B in network II happen to share the same address, yet they remain oblivious to this fact as long as the two networks are not joined.

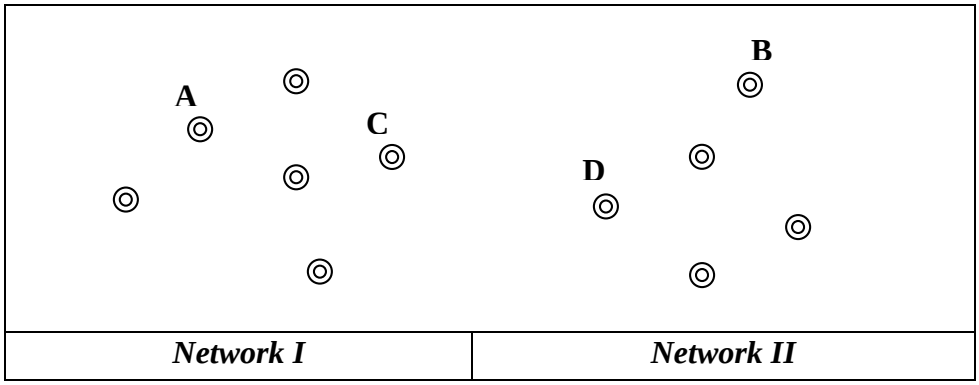


Figure 2.2: Example of Two Separate MANETs

Now consider a case where the networks merge, e.g. if node C moves into the range of node D. The presence of two nodes, A and B, with the same IP address will cause incorrect routing.

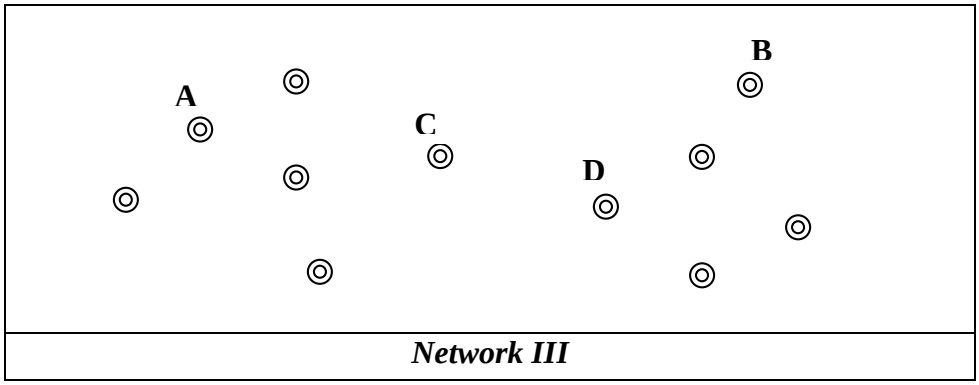


Figure 2.3: Example of Two Merging MANETs

In order to handle such topology changes, the addressing scheme needs to either be stateful (such as the one proposed by Sun et al.) or rely on another form of node identification (in a manner similar to Weak DAD). However, the Weak DAD addressing scheme requires close incorporation with the routing protocol. The routing protocol should consider the additional node identifier as well as the IP address to identify a route. This may cause some impediment in the choice and implementation of the routing protocol. But since both the addressing scheme and the routing protocol will be implemented, Weak DAD seems to be a good choice. It already satisfies the first three properties we desire, just as DAD does. In addition, it handles network merges and partitions, satisfying the fourth desired property.

2.3 Routing

No research area in mobile ad hoc networks is more active than routing. A large number of routing protocols have been proposed over the last few years, and this section provides an overview of these protocols by classifying them into three different categories. The following section highlights the routing requirements of the target scenario, according to which, a small set of protocols are proposed for the project. These protocols are then discussed in a little more detail.

2.3.1 A Review of the Protocols Available at Present

There is a large number of routing protocols for MANETs that have been developed over the years [45]; hence it was impossible to go through all of them within the time frame of the project. In addition, it might not have been fair to select a few of these protocols and then compare them to each other.

Instead, we have aimed at considering all protocols by classifying them, and then choosing the class which best serves the demands of our target problem domain. Then, we compared protocols from the chosen class with each other in a simple simulation which has characteristics parallel to those of the target network.

The MANET routing protocols can be classified as follows:

- Reactive / Proactive
- Location-aware / Location-unaware
- Hierarchical / Flat

Proactive protocols, sometimes referred to as *table-driven* protocols, behave like most routing protocols designed for wired networks. They aim to establish a consistent view of the network at all nodes by using one table or more to hold routing information. The nodes exchange updates about the topology periodically and/or when the topology changes. This way, all nodes know at least one route to all other nodes within the network at any given time. Examples of such protocols include DSDV (Destination-Sequenced Distance-Vector Routing protocol) [28], WRP (Wireless Routing Protocol) [24], and Guesswork [26].

It can be argued that proactive protocols are not suitable for mobile ad hoc networks on the basis of bandwidth and processing overhead. Regular updates threaten to exhaust the available bandwidth, which is a relatively valuable resource in wireless networks. As the number of nodes in a network increases, the total number of updates increases as well. Moreover, as the network grows, a simple change in topology will spawn updates from a larger group of nodes. This overhead does not only affect the bandwidth utilization, but also imposes more processing overhead upon all nodes. Such an overhead should be avoided at all costs, in light of the limited processing capability and finite power supply of most mobile nodes.

On the other hand, reactive protocols provide a route between source and destination nodes only when such a route is needed for transmission. This is why reactive protocols are also called *on-demand* protocols. Using such protocols, the transmitting node requests a route to its destination. The request is propagated throughout the network until a reply is received either from a node which knows a route to the destination or from the destination node itself. The route is then maintained by control messages until it is no longer desired. These control messages are similar to routing advertisements, except that they are only exchanged between the sender and the receiver, rather than being broadcasted, and that they are triggered by route changes.

Although reactive protocols require far less connection maintenance overhead than proactive ones, they obviously increase the connection setup cost. Nevertheless, two of the most studied routing protocols for ad hoc mobile networks happen to be reactive ones: AODV (Ad Hoc Distance Vector) [27] and DSR (Dynamic Source Routing) [19]. These two protocols have been designated by the IETF MANET

Charter (previously the IETF MANET Working Group) [40] as experimental standards. Examples of other reactive routing protocols are TORA (Temporary Ordered Routing protocol) [25], ABR (Associativity-Based Routing protocol) [34], and SSR (Signal Stability Routing protocol) [15].

A location-aware protocol uses the location of the nodes to provide a more efficient and optimized route. Location information is retrieved from systems integrated within the mobile device, such as a GPS chip. In contrast, location-unaware protocols do not take the location of the devices into consideration for route optimization. Instead, this class of protocols uses physical properties related to the links (such as link bandwidths, link reliability, hop count, etc.) rather than the nodes.

Routing protocols can also be classified according to their clustering techniques. Although most protocols do not, some protocols divide the network nodes into different clusters of nodes. Dimakis et al have developed a scheme where routing is performed either inside a cluster or in between clusters [14]. CEDAR (Core-Extraction Distributed Ad hoc Routing) [32] selects a set of nodes to form the core of the network. The core nodes make up a self-organising infrastructure that uses link-state updates to allocate resources for traffic across the network. Other examples include CGSR (Clusterhead Gateway Switch Routing protocol) [9]. Such hierarchical architectures may manage to reduce the management overhead only in cases of very large mobile ad hoc networks.

2.3.2 The Routing Requirements in the Target Scenario

A device's *neighbours* can be described as the devices which reside within its range of wireless connectivity. When a new neighbour is discovered, a node will start to exchange information about its own neighbours and vice versa. After the exchange, a device will have knowledge of the surrounding devices, including at least their IP addresses and unique identifiers (which can be the IP address itself or a unique key).

Once the user decides to forward a message, his or her device will start requesting routes to the surrounding devices. Upon receiving route information, a connection will be established for the application to handle the forwarding of the message. The application should start to exchange message metadata. According to this information, none or some of the messages will be exchanged. If any messages are to be exchanged, transfer commences. Otherwise, the session ends and each of the nodes look for other nodes which might have interest in message exchange.

At first glance, it may seem that messages need only traverse single-hop routes. In that case, there would be no need for a routing protocol at all. All that is needed for any particular node is merely to announce its presence, which it already does according to the 802.11 specifications, and then other nodes in its range can directly communicate with it. However, in the real world, different users would be interested in different types of messages. For instance, consider three devices A, B, and C shown in Figure 2.4. The Figure shows the transmission range of each device. Devices A and C are out of each other's range and hence cannot directly communicate except through an intermediary node B. Consider also that device A holds a message that might be of interest to user C but not to user B. If only single-hop routes were to be used, such a message would never reach C except if it can be directly reachable from A.

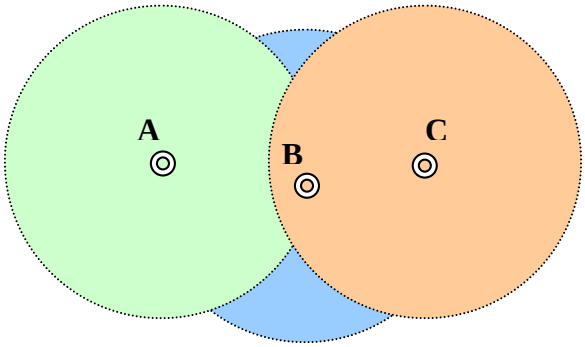


Figure 2.4: Example to Illustrate Single-hop Routes

Such an approach relies heavily on the incentives given to users to persuade them to accept and forward messages with less attention given to the relevance of the message content. This approach was supported by the project's industry link. Nevertheless, we have chosen to implement some routing functionality to help messages propagate beyond a single collision domain. This makes the work suitable for situations where the incentive system fails, or where it does not exist at all.

2.3.3 Choosing a Routing Protocol That Fits the Target Scenario

If we were to use a reactive (on-demand) protocol, then each node would only communicate with those nodes which are within its range. No routes would be requested for distant nodes until they are known through exchange of routing information. However, this would not be the case should we use a proactive (table-driven) protocol. In the latter case, each node will have enough routing information about all the nodes in its network. Such consistent view of the network, as explained earlier, is established at a high bandwidth and processing overhead.

In the mobile devices of our network of customers, there is no urge to impose a routing protocol that keeps a consistent view of the network on each device. Firstly, we need to keep the span of the message propagation to one hop at a time. Our goal is to propagate messages from one device to others. Each user will be encouraged to help in propagating the messages. However, nodes should not use their neighbours as routing agents to get to other nodes which are out of their range. For example, consider source node A, destination node C, and node B which is a neighbour to both nodes A and C. If C is out of A's range, just as illustrated in Figure 2.4 above, A should not use B as a relay to get to C. Instead, A should exchange the message(s) with B, and then B will forward it/them to C. In other words, if A cannot reach C except through B, then A should forward to B and then B can forward to C. This way both A *and* B get rewarded for forwarding. However, if C happens to be in A's range, A can send to it directly or can use B as a relay if this route offers more throughput or less delay. Keeping track of this using a proactive routing protocol would be impossible, since A would already know a route to C. The message(s) will only get as high as the network layer in node B. Hence, the message(s) will go undetected by B's application layer, and B will not be rewarded.

The second reason why we do not want to keep a consistent view of the network at each node is that the network itself might not be consistent. Users of the system are expected to relocate frequently, by walking around a great deal of the time. In such

conditions, high broadcast traffic is a high price to pay for a topology that may not necessarily last long. Thus, it is more practical to use an on-demand routing protocol. In fact, we would rather have a routing protocol that performs better when the network is constantly changing. Additionally, the bandwidth overhead created by proactive protocols would be overwhelming as the network scales beyond a few hundred nodes. The processing overhead will also be high, consuming a considerably large proportion of the power of the devices. It is for all these reasons that a reactive protocol is favoured over any proactive protocol.

As mentioned before, location-aware protocols help find the most efficient route between source and destination nodes. This added sophistication, however, is only available with special hardware installed in all devices. For lack of such a guarantee, location-aware protocols are ruled out.

Hierarchical or clustering protocols provide a significant improvement in routing overhead only in large networks which are of a relatively stable topology. Our aim is to provide an application deliverable in networks which are expected to constantly change.

In conclusion, we shall be only considering reactive, location-unaware, flat routing protocols. Protocols that fall under this classification are AODV, DSR, and SSR among others. We have chosen to focus on DSR and AODV as they are recognised as experimental standards and as two of the most efficient MANET routing protocols.

2.3.4 Dynamic Source Routing (DSR)

Using DSR, each node keeps a cache of the most recently used or learnt routes [19]. As the name suggests, the DSR protocol is based on source routing approach. When a connection needs to be established with a destination to which a route is not known, a *route request packet* is issued. This packet is broadcasted to all neighbours. Upon receiving a route request packet, a node will look in its cache for a route to the requested destination. If a route is found, a *route reply packet* is sent back to the inquiring node. Otherwise, the node will add its own address to the hop sequence in the route request packet, and then forward it to its neighbours.

The route reply packet contains a record of the complete route in the form of the accumulated hop sequence. However, the reply must not use the same path the route request traversed. The replying node follows the same procedure to find a route to the route request packet originator. If a route is found in its cache, then it is used. If not, a new route request packet is piggybacked to the outgoing route reply packet. Although this reverse route discovery increases the overall routing overhead, it allows finding two different paths for the same connection. As a matter of fact, a route cache may contain several routes to the same destination. This feature makes the DSR protocol suitable for networks with unidirectional links. It also facilitates balancing the load of one connection over more than one path.

Routes are maintained in DSR using acknowledgements and *route error packets*. Each node that forwards a packet sends an acknowledgement back the original sender. If an acknowledgement is not received from an intermediate node, then a route error packet is sent to all nodes using this route. Consequently, the corresponding routing

cache entry is invalidated. All other entries for routes passing via the faulty node are truncated.

2.3.5 Ad hoc Distance Vector (AODV)

AODV is a routing protocol with a minimalist on-demand tactic; it is only used when a node does not have a route to a destination [27]. It can be described as a pure on-demand routing protocol. AODV keeps the routing tables at each node at an absolute minimum, requiring a fresh route to be discovered for each new connection. Each routing table entry has an associated expiration timer. Although this approach might seem to induce more control traffic in setting up connections, the added network overhead is not that significant in our target scenario. In this target scenario, any connection between two particular nodes is established only once for a relatively long period of time. The same connection is only established again after some time when there are new messages to exchange.

AODV uses three control messages (sent over UDP). A node's routing table readily contains entries to all its neighbours discovered using beaconing (a mechanism part of the IEEE 802.11 specification [1] which is explained in Appendix B). If a connection needs to be established with a destination for which there is no matching entry in the routing table, the node issues a Route Request (RREQ) packet and broadcasts it to all its neighbours. The issued RREQ packet contains a newly generated destination sequence number, which is similar in concept to a timestamp. Each receiving node will first use the RREQ packet to update its routing table with a route to the source node. Then, it will either forward the RREQ packet to its neighbours, or reply to the source node with a route if a 'fresh' one is found in its routing table. A fresh route is defined as one with a destination sequence number larger than the one contained in the RREQ packet. The reply packet is referred to as a Route Reply (RREP) packet and is sent to the source node through the same route the RREQ packet traversed.

The third AODV control message is the Route Error (RERR) packet. When a node finds a path to a neighbour invalid, it will issue an RERR packet to only those nodes that had active connections with the dead neighbour. Each node maintains a list of the nodes with active connections to its neighbours. Upon receiving a RERR packet, a node invalidates the corresponding route table entry.

One feature of AODV which is often considered a drawback is that it restricts reply packets, when returning to the sender, to the same route used to reach the destination. In other words, AODV cannot operate in networks where unidirectional links exist. It also means that traffic between any pair of sender and receiver cannot be balanced on more than one link. This aspect of AODV is acceptable as we do not anticipate deploying the application in networks with unidirectional links, nor is it an aim for us to achieve load balancing over multiple routes.

2.3.6 Comparing DSR and AODV

In order to settle the comparison between DSR and AODV, we have simulated a few MANETs, each under a number of conditions that portray the target scenarios. In these simulations, we have used the DSR, AODV, DSDV protocols. An additional simulation model with no routing protocol was also used. We have extracted the DSR and AODV results from two of the simulation runs to display them here in form of graphs. Full results and analysis of the simulation can be found in the Appendix C.

The first simulated network consisted of 20 nodes. Node movement and TCP flows were generated at random and stored before the simulation. This was to ensure that the simulation of each routing protocol uses the very same network. The results of the simulation can be summarized in the following graphs (figures 2.5, 2.6, and 2.7).

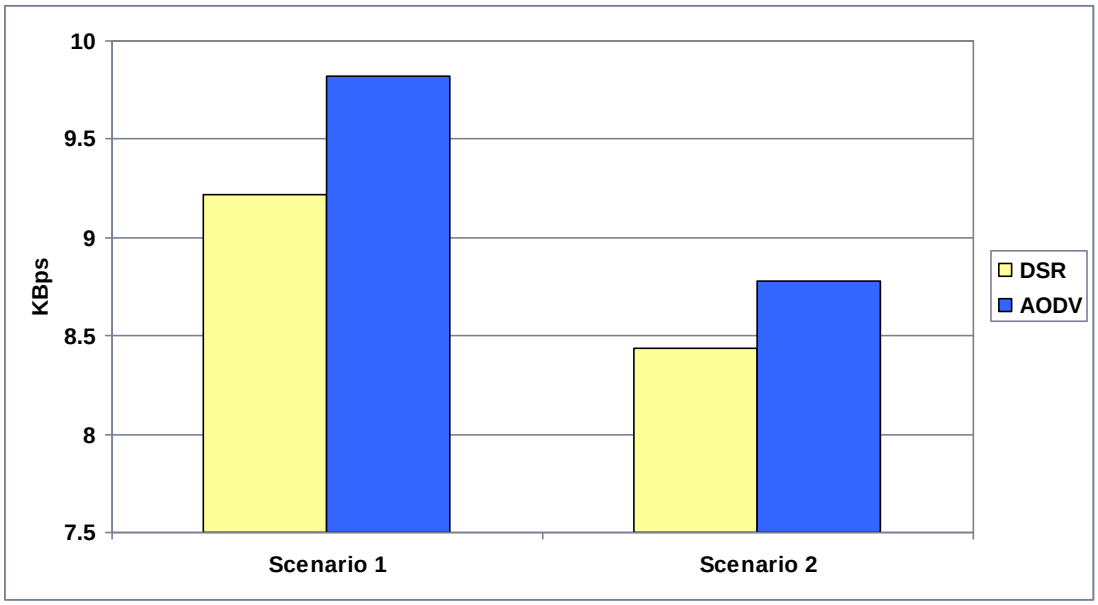


Figure 2.5: Average Throughput of DSR and AODV for 20 Nodes

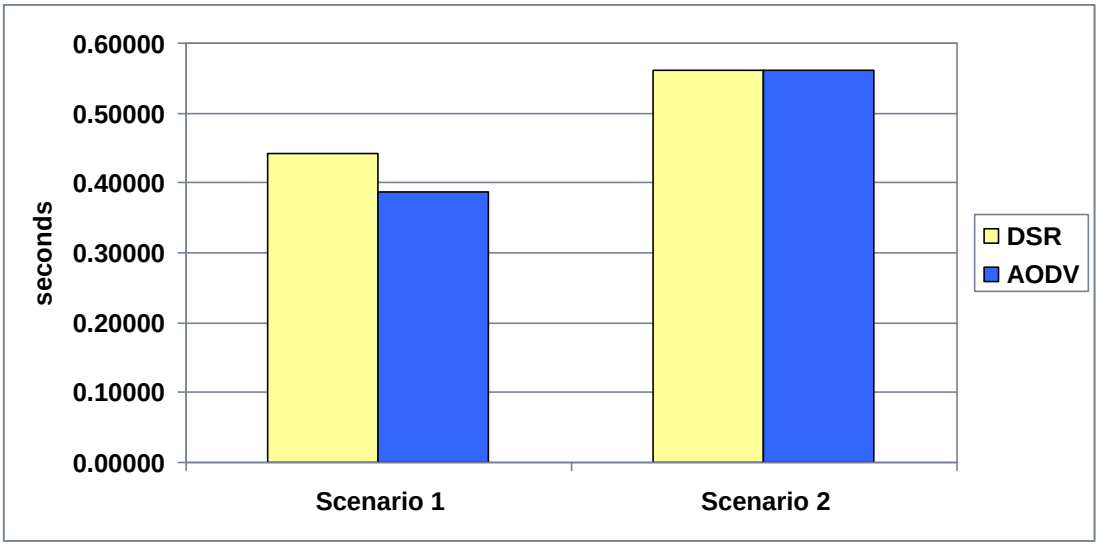


Figure 2.6: Average End-to-end Delay of DSR and AODV for 20 Nodes

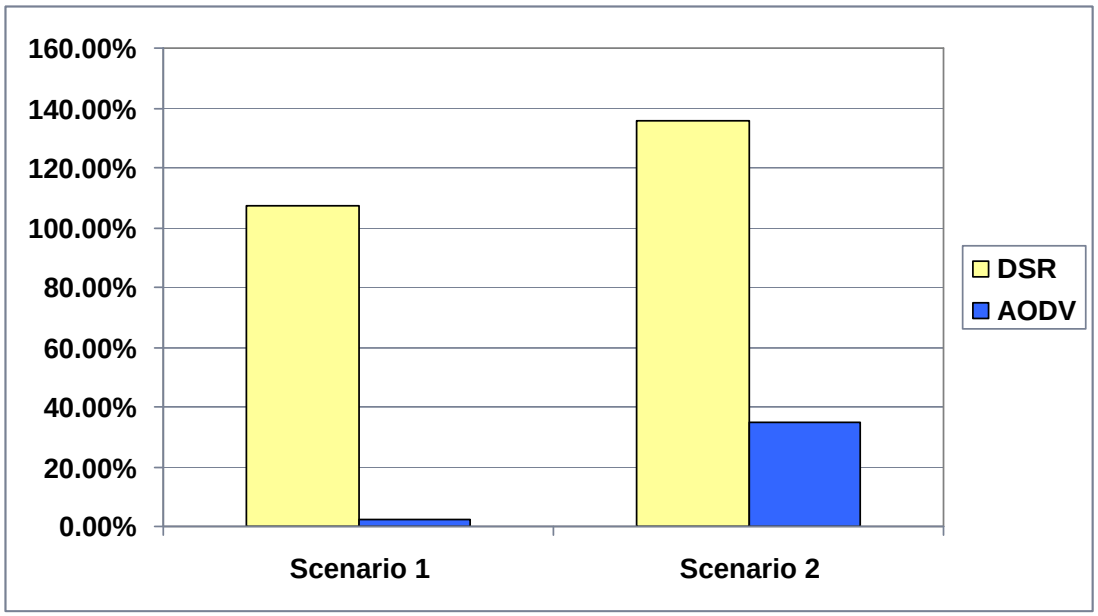


Figure 2.7: Routing Overhead of DSR and AODV for 20 Nodes

All scenarios used the same generated traffic between the nodes, but differed in the manner in which the nodes moved. Scenario 1 used nodes moving at a speed between 0 and 2 m/s with an average pause period of 0.5 seconds between the movements. Scenario 2 had nodes moving at a random speed between 0 and 1 m/s with an average pause period of 1 second. Scenarios 1 and 2 are referred to as scenarios II and IV in Appendix D, respectively.

The values of these parameters might seem unrealistic. The intention was to create highly dynamic networks where nodes enter and leave at a high rate. Ideally, we would want the application to handle such unstable environment without degradation in the service it provides and with the minimum induced overhead.

It is clear from the graphs that AODV provides a higher average data throughput, at a radically lower routing overhead. The DSR overhead is relatively high due to the fact that it is based on source-routing, i.e. each routing packet contains full routing information which keeps accumulating as the packet propagates through the network. AODV packets, on the other hand, are only comprised of the destination node address.

A similar simulation has been carried out with a slightly larger network of 50 nodes. Figures 2.8, 2.9, and 2.10 illustrate the results of this simulation:

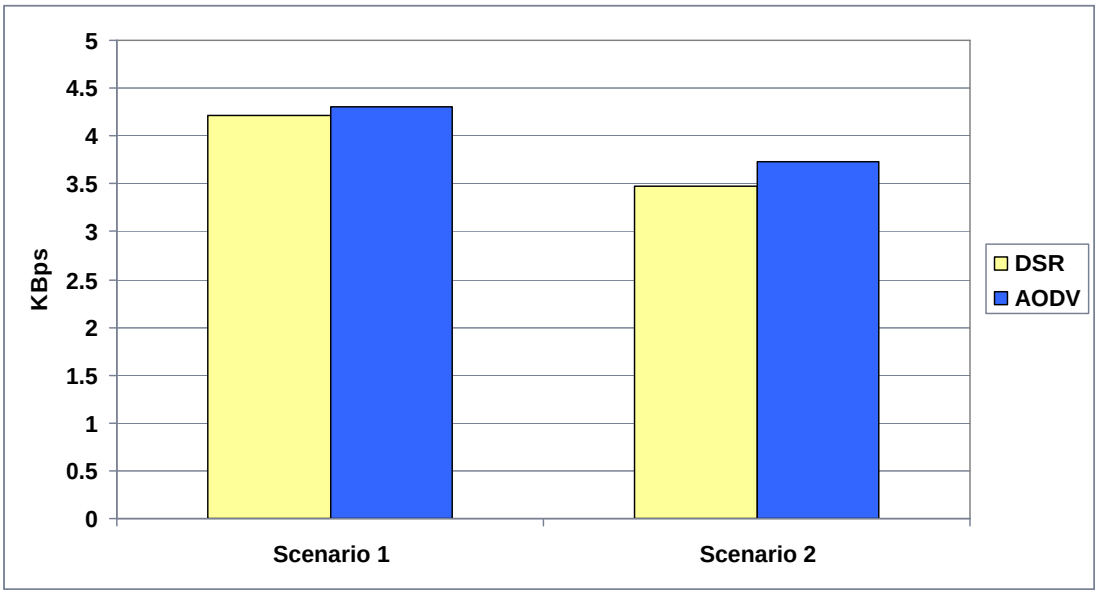


Figure 2.8: Average Throughput of DSR and AODV for 50 Nodes

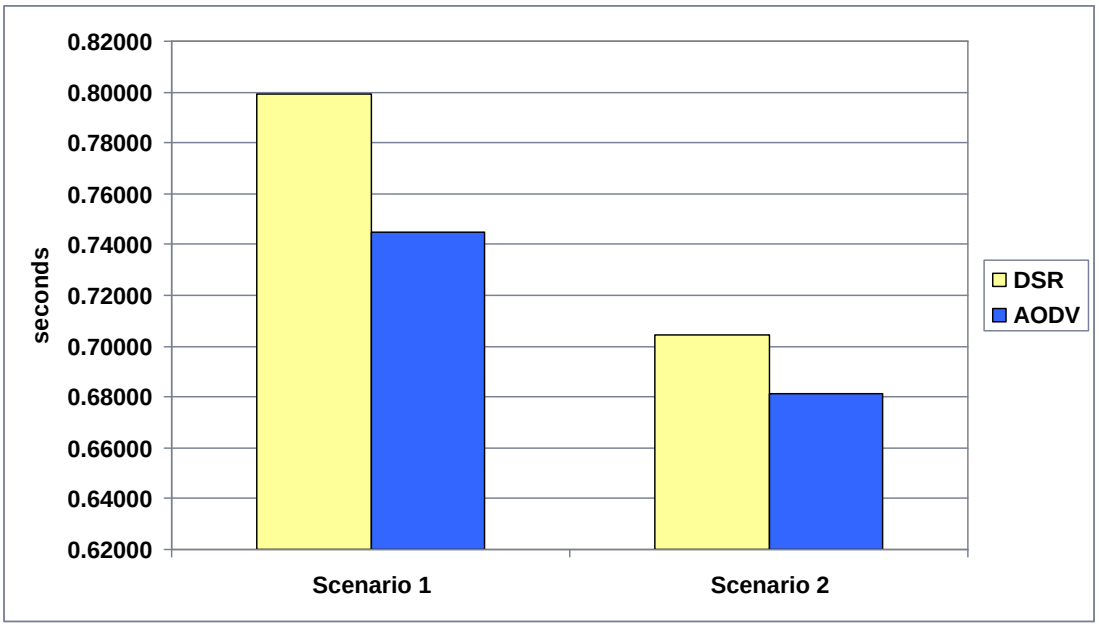


Figure 2.9: Average End-to-end Delay of DSR and AODV for 50 Nodes

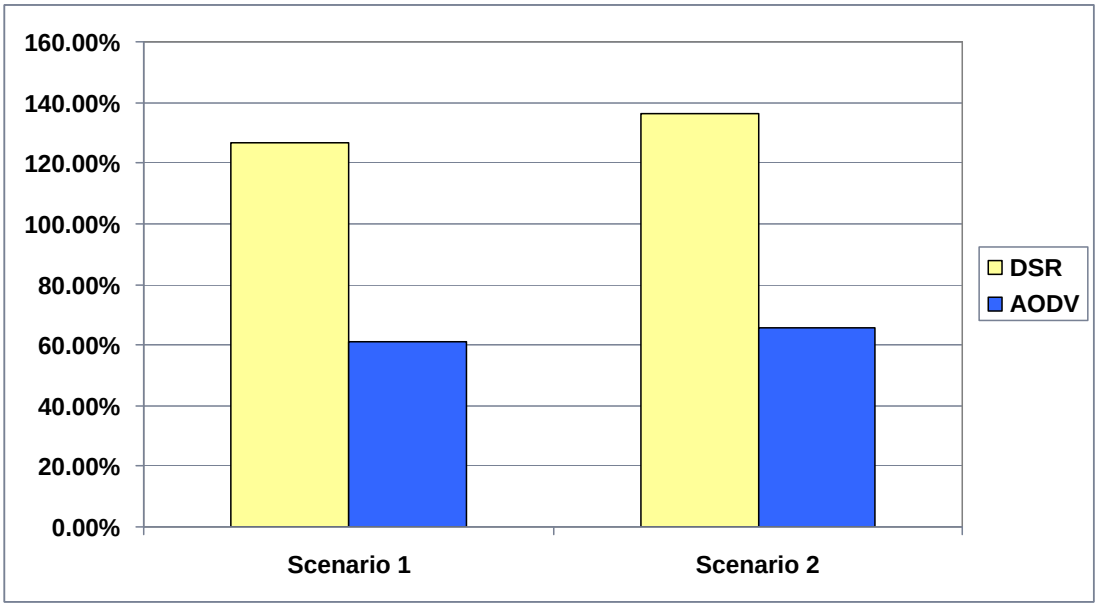


Figure 2.10: Routing Overhead of DSR and AODV for 50 Nodes

AODV still offered higher throughput as well as less delay and less routing overhead as the network scaled to 50 nodes. AODV has also proved to display better results in a simulated network of 100 nodes. Please refer to Appendix D for full details of the simulations.

2.3.7 Conclusion

We have classified MANET routing protocols from RFCs and reports using three different classifications. For each of these classifications, we have chosen the ones that would serve our target application. Those were reactive, location-unaware, and flat routing protocols. From the protocols that fall into these classes, we have chosen the two protocols which are recognised by the IETF MANET Charter [40] as experimental standards. Then, we evaluated the performance of these two protocols in a simple simulation. The results from these simulations have proved that AODV induces much less overhead than DSR, as well as lower average end-to-end delay and higher average throughput. Using these simulations as a fair and relevant comparison, we have decided to choose AODV as our routing protocol.

2.4 Related Work

There is a huge interest in MANETs from both the research and business worlds. In this section, we touch on a few examples of other work which is concerned with the same issues dealt with in this project. These examples should draw a picture of the huge potential of MANETs, and they should help realise the importance of this work.

British Telecom is also sponsoring another M.Sc. project within the Computing Department at Lancaster University. The said project, prepared by my colleague Dixita Patel, focuses on the implementation and evaluation of the higher layers of the same target application. The project tackles issues such as security, accounting, user profile matching, and messaging policies. Accounting information is sent to the provider's network via GPRS, hence the title of both projects.

The motivation of exploiting the potential of the WiFi market has driven a large number of projects beside this one. The range of applications includes personal area networks (PANs), military networks, sensor networks, and vehicular networks. The most prominent example is probably the DAIDALOS (Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimised personal Services) project. This project aims to provide pervasive wireless services by the integration of heterogeneous networks. One of the main objectives is to utilise ad hoc networks by gracefully integrating them with 3G (third generation) networks of mobile providers. The project started in 2002, and is now in its second phase. DAIDALOS phase 2 is planned to conclude by 2008, and has a budget of €22.1 million [50].

There are a number of projects working on using MANETs in the area of inter-vehicular communication. These projects aim at extending the range of awareness of motorists by exchanging event updates, such as traffic information and road conditions, between vehicles using WiFi. The Car2Car Communication Consortium is a non-profit organisation established by six major European car manufacturers [49]. It aims at delivering an industry standard for inter-vehicular communication. One of the projects delivering these standards to the Car2Car Communication Consortium is NOW (Network On Wheels) [58]. The project tackles issues such as communication and security in MANETs. SPAMM [59] and FleetNet [53] are similar projects.

Moreover, there is also a large number of projects, mostly commercial, working on building applications for PANs. One example is the Zune project by Microsoft [69]. Zune is a portable audio player equipped with a WiFi adaptor. The hardware is contributed by Toshiba, while Microsoft is working on providing the application that will allow Zune owners to stream audio files to other users in the same MANET. Sony has previously uncovered its plans to provide a similar product, Mylo. Although Mylo (My Life Online) is to have a few more features than Micorosoft’s Zune, such as a fully-supported web browser and Skype (for cheap VoIP calls) [71], the ad hoc functionality is still limited to streaming files.

The prospects of employing MANETs do not stop at commercial applications. Several projects aim at studying the autonomy of MANETs in order to employ them in extending infrastructure networks. ANA (Autonomic Network Architecture) is a joint project between Lancaster University and several other academic and commercial European institutions [47]. It aims at expanding the Internet beyond its traditional technologies by incorporating resources in the form of mobile devices in ad hoc wireless networks. The goal is a network that adapts itself according to its resources and the demands it is expected to meet. Haggle is another project, initiated by Intel and Cambridge to manage and share resources across MANETs formed by personal WiFi-enabled devices [55]. As described on the project's homepage, the project corresponds to the “ad hoc Google”. Needles to say, there is a strong connection between the ANA and Haggle projects, and the project proposed by this report: message propagation forms the basis of the approach to organise MANETs.

Several other work groups are concerned with the services that can be provided by integrating MANETs into other communication environments. DTNRG (Delay-Tolerant Networking Research Group) [51], E-NEXT (Network of Excellence Emerging Network Technologies) [52], BIONETS (BIOlogically inspired NETwork

and Services) [48], MobiLife [57], IP CASCADAS (Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services) [56], and FOKUS [54] are some examples of such work groups.

2.5 Conclusion

So far, we have defined the goals of the target application, and analysed the main concerns that affect the outcome. We then conducted a survey of recent research regarding these main concerns, particularly the issues of addressing and routing. From this survey, we have reached decisions on addressing and routing in our target network. Finally, we have listed some of the extensive work which is related to that carried out in this project.

3 Design

This chapter discusses the messaging protocol that has been developed to run on top of AODV in a simulated environment. The protocol is first discussed in terms of what it can and cannot do. Then, the major design decisions affecting the performance of the protocol are introduced and discussed in section 3.2. Section 3.3 gives more details about the messaging protocol, concentrating on how data flows in the system.

In this chapter and the next, the propagated content will be referred to as “adverts” instead of messages. This is to avoid confusion with other control messages introduced later in this chapter.

3.1 Overview

This section introduces the functionality of the messaging protocol.

3.1.1 Aims

The application aims to:

- Inform users of the adverts available in the network.
- Allow the exchange of adverts between users.
- Maintain a cache of adverts, deleting any which expire and saving received ones.
- Prevent old or duplicate adverts from being exchanged.

3.1.2 Shortcomings

As described in section 2.1.5, the application is only a simple realisation concerned directly with the exchange of adverts. More work is expected to be integrated into the application before it can be deployed as a complete solution. This extra work includes a few modules and minor data structure changes. However, it is important to state that the following features are not implemented in our application:

- The insertion of an advert into the network through a wireless access point.
- Security measures to prevent SPAM adverts.
- Feedback that credits each user for forwarding an advert.

Nevertheless, we shall point out, in the course of illustrating the application, where such features can be integrated.

The messaging protocol runs above IPv4 but just below the transport layer. The protocol is connectionless-oriented, using separate messages instead of flows. This is mainly due to the fact that connection-oriented transport protocols, such as TCP, are not suitable for wireless networks where packet loss is a common phenomenon. For this reason, we have chosen to create our own header. This header will be attached to the IP packet.

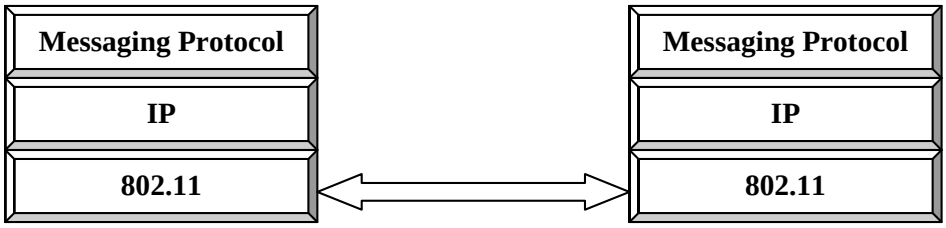


Figure 3.1: The Protocol Stack of the Mobile Devices

3.2 Major Design Issues

In section 2.3.3, we have established that nodes need not keep a comprehensive view of the whole network. This applies to both the knowledge of routes and the knowledge of adverts in the whole network. Thus, our application should be based on mutual exchange of adverts rather than network-wide exchange.

One of the requirements of the messaging protocol, as illustrated in section 3.1.1, is to keep track of the exchange of adverts. In order to establish this, every advert has to have its own unique integer by which it can be identified. Adverts also contain some metadata that describes their content. To prevent adverts from propagating for an indefinite amount of time, each is tagged with a timestamp after which it expires.

Nodes should keep a summary of the adverts they currently store. This summary will be broadcasted to other nodes in the network. If interested, these nodes would request one or more of the specified adverts. However, a primary goal of the design of the messaging protocol is to reduce the following three costs: processing overhead, required memory space, and network overhead.

3.2.1 Processing Overhead

The more frequently the application updates the summary of stored adverts, the more processing power it demands. It is not one of the main goals of the project to design an application which conserves the device's battery power as much as possible, yet it is only sensible to take this aspect into consideration. After all, no one would like to implement an application on a mobile phone when the application is using up more power than seems necessary for the kind of service it provides.

3.2.2 Memory Space

A summary (list) of the stored adverts is maintained and stored in a data structure. The fields of this list can include the advert identifiers only, keeping the list to a bare minimum. Alternatively, the list can include more fields, such as the title of the advert and the type of content it comprises (e.g. clothes, sporting goods, holidays, etc.). The choice cannot be made without considering the network overhead (section 3.2.3) since the network overhead is dependent on the size of the list.

3.2.3 Network Overhead

The magnitude of the network overhead created by the application on one node is the product of two factors: the size of the information which is to be exchanged (discussed above), and the frequency of exchange. If the nodes were to exchange brief information about the adverts, then that would induce more messaging. Consider the option mentioned in the previous section where the data structure contained only the advert identifiers. When a node receives such a data structure, it would first compare the identifiers to determine which adverts are new, and which are not. Then it would request more info about these new adverts, such as titles, etc. Once such information is received, the transfer of such adverts can be accepted.

The other extreme is to send the complete adverts instead of the metadata. Obviously this solution is not practical, especially if the adverts contain are more than text. It would induce very high traffic into one collision domain which would, eventually, cause the network to collapse.

3.2.4 Conclusion

Regarding the processing overhead, it was decided to reduce the frequency of scanning the stored messages and to update the data structure. Instead of performing this scan process periodically, we have chosen to perform it only when a new message is received. During this scan, expired adverts (according to their expiry timestamp) are removed and the data structure is updated accordingly. If no new adverts are received, then no scan is done and hence old adverts are not removed. This is not a major problem as there is no need to clear the space if no new adverts are received.

It was also decided to pick a moderate point between the two extremes regarding the size and frequency of the information exchange units. Referring back to the second assumption in section 2.1.5 regarding certain functionalities which were not to be implemented in this project, profile matching is one of these functionalities. In our simplified version of the application we shall only include the identifier and the title of the advert as fields of the data structure summarising the adverts acquired by a device. Additional fields (more specific advert metadata) should be added to implement additional features such as profile matching.

Even with a few more metadata fields added to the list, the size of the data exchanged is not considerable. This allowed the use of broadcasting to inform other nodes of the list. The wireless medium is by nature a broadcast medium. Hence, broadcasting the list would not cause any added network overhead over unicasting it. It is the frequency of the broadcast, however, that would determine the magnitude of the added network overhead.

3.3 Protocol Design

We shall now explain how the application manages the propagation of adverts as defined by the design above. It should be noted that in this implementation, we assign each advert with its unique 32-bit long identifier, although in a real implementation such an identifier would probably be assigned by the message issuer (i.e. the provider).

Each node stores the adverts that belong to it in the available memory (phone or external memory card/stick). In case a new advert is added or an old one expires, the

node creates a new *Catalogue*. The Catalogue is a simple data structure consisting of a list of the adverts stored, and a timestamp of when the Catalogue was created. The Catalogue will later be used in the exchange of adverts with other nodes. Figure 3.2 illustrates the fields of a Catalogue entry.

Advert_ID	Title

Figure 3.2: Fields of a Catalogue Entry

Catalogues received from other nodes are kept in a table called *Catalogue Record* (CR). Each entry in this table corresponds to a node in the MANET. An entry that corresponds to node A, for example, contains A's identification and the timestamp of the most recent Catalogue received from A. The Catalogue Record table is represented in Figure 3.3.

Node_ID	Newest_Timestamp

Figure 3.3: Fields of a Catalogue Record Table Entry

Each node sends out its Catalogue on a periodic basis to the IP broadcast address with a maximum hop count (TTL, time to live) of 1. All nodes in the direct range of the broadcasting node, i.e. its neighbours, will receive the Catalogue. On receipt, every neighbour takes two actions: it updates its Catalogue Record table, and it replies with a *Handshake Message*. If node A, for example, was to send a Handshake Message to a node B, the message will contain the current Catalogue of A, its timestamp, and the timestamp of the most recent Catalogue received from B (obtained from the Catalogue Record table).

Catalogue	Catalogue_Timestamp	Newest_Timestamp

Figure 3.4: Fields of a Handshake Message

Consider node A that has just broadcasted its Catalogue. One of its neighbours, node B, receives it and checks its Catalogue Record for an entry of A. If none is found, a new one is created with 0.0 as the value of the timestamp of the most recent Catalogue. If an entry already exists for A, it is deduced that A has rejoined the network after being out of reach for a brief period of time. Then, B will compare the value of the received Newest_Timestamp to the timestamp of its current Catalogue. If they are different, B will reply to A with its own Handshake Message. This exchange updates each node with the adverts possessed by each of them as illustrated in Figure 3.5. Next, device B prompts its user with the contents of the received Catalogue if it contains IDs of new adverts. The user can then decide to receive all, some or none of the adverts that A holds. According to the user's choice, the transfer of data can commence. It is here where the received advert should be checked for authenticity. It is also here where a module should be implemented to keep track of forwarding credits and accounting.

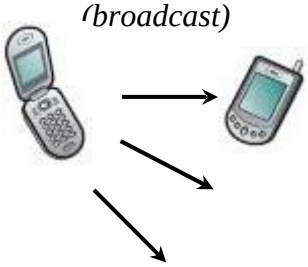
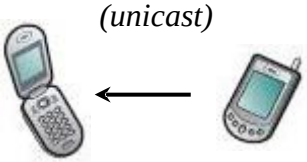
Device A broadcasts its Catalogue periodically ⇓ *A broadcasts its current Catalogue within a Handshake Message (Newest_Timestamp = 0)*	**A** **B** 	
	Handshake Message (broadcast) 	*B updates its CR table with the new Catalogue of A* ⇓ *If A has an old Catalogue of B, then it will be sent the latest one in a Handshake Message* ⇓ *Device B will prompt its user about the new adverts A has (if any)*
A updates its CR table with the new Catalogue of B ⇓ *Check that B has the latest Catalogue of A* ⇓ *Device A will prompt its user about the new adverts B has (if any)*	*Handshake Message (unicast)* 	

Figure 3.5: The Exchange of Catalogues

When an advert is received, a node will go through the following sequence of actions:

- Scan through the cache of stored adverts, deleting any expired ones.
- Add the new advert to the cache.
- Update the Catalogue.
- Broadcast the new Catalogue, and restart its timer.

3.4 Alternative Approach

Initially, we have thought of using notifications from the routing protocols instead of periodic broadcast messages in order to reduce induced traffic. Using this technique, we would need to modify AODV to inform our application whenever a new node is discovered. Following such an event, the node is triggered to send a Handshake message to the newly discovered neighbour. Such notification will be in the form of an *upcall*.

In addition to the difficulty of going through AODV code and modifying it at the risk of creating misplaced upcalls, this approach would not work due to the fact that AODV is an on-demand ad hoc routing protocol. AODV would not look for

neighbours but would issue a route request upon connection initialisation. For this method to work, simulations had to include background traffic that would cause nodes to generate AODV route requests. However, neighbour discovery in this case would rely heavily on background traffic and would not yield trustworthy simulation results. Moreover, it could not be assumed that the target scenario would have background traffic by default. For these reasons, this approach was soon aborted.

It should be noted that this approach requires periodic scanning of the Catalogue Record table to ensure that it is up to date by removing old entries.

A similar approach would be to use MAC upcalls. A Handshake message would be issued whenever the MAC layer detects a neighbour through 802.11 beacon messages (please refer to Appendix B for more info about the operation of the 802.11 MAC layer). This approach would not necessitate background traffic, yet, it may be difficult to implement and debug since most of the MAC layer code runs in kernel space.

3.5 Conclusion

Chapter 3 has reviewed the functionality of the messaging protocol. It has also presented three issues that were taken into consideration in designing the protocol. Then, the different control messages and data structures of the protocol were defined with focus on how data flows in the system.

4 Implementation

While the previous chapter has outlined the design of the messaging protocol, this chapter explains how the protocol was implemented to run on the simulation environment. First, however, we shall describe how we have chosen which simulation environment to use. Then, in section 4.2, we give an overview of the chosen simulation environment. Section 4.3 elaborates on how the messaging protocol was implemented. Section 4.4 briefly discusses the process of integrating our protocol in the simulation package. Finally, the chapter concludes with comments on our implementation.

4.1 Choice of Simulator

Network simulation is a technique used to observe, evaluate, and improve the communication in networks without the cost of real deployment. The behaviour under different conditions in a simulated environment should reflect the behaviour under the same conditions in an actual network.

The importance of simulators arises from the compared difficulty of deployment in a real network. In addition to the high cost of providing a real large-scale network, debugging can be a cumbersome task. Deployment on a simulated network is much simpler, making it easy to improve and revalidate the protocol repetitively. Network simulators also allow easy metrics calculation, such as the number of packets or the packet processing time.

There is a variety of different network simulators, the majority of which have been developed towards networks in general. There are a growing number of simulators, however, which are targeted towards networks of special nature like MANETs. The following is a rough outline of simulators which can be used to evaluate the project.

The choice of simulator depended on a number of factors. Firstly, the simulator should fully support wireless protocols and standards. Secondly, it should provide enough flexibility in order to allow easy modification to protocol implementation even at low layers, and to specify custom network parameters. Third of all, the simulator should be of previously proven results. Moreover, support for using the simulator should be available. Also, the simulator is expected to collect detailed network metrics. In addition, it is preferred that the simulator would be capable of providing a real-time graphical visualisation of the network.

First-hand experience was used to make an informed decision regarding which network simulator to use. A simple demo was developed and tested using each of the simulators mentioned below. The demo consisted of a topology of 5 nodes which are

interested in establishing connections with each other for a short period of time. The nodes were setup to simply move at random directions and speeds.

ns2 [2] [42] is a popular open-source simulator widely used for academic and educational purposes. The different OSI layer protocols are implemented in ns2 as separate modules. This facilitates integrating a customised protocol into the stack. ns2 is event-driven, and it supports both wired and wireless networks. It is written in two languages: C++ is used for the core engine, while OTcl [44] is used to configure the simulation. OTcl is an object-oriented version of the TCL (Tool Command Language) which is an easy-to-use programming language widely employed in prototyping. ns2 is accompanied by nam, a tool used to animate the simulated network and display statistics.

One of the key features of ns2 is the abundance of information on it. The developers have produced a very thorough manual. The official ns2 website also provides a mailing list for exchange of knowledge and experiences. Moreover, bugs are fixed and more functionality is being added on an almost daily basis.

OPNET [64] is one of the principal simulators, mainly used to test and evaluate solutions for commercial, government, and military networks. However, OPNET is not free to use and is hence not an option.

GloMoSim [60] was developed initially at UCLA Computing Laboratory. Although GloMoSim provides a scalable network simulation environment for wireless networks, the flexibility of the architecture of the simulated network is quite restricted. Moreover, we have decided not to go with GloMoSim for the lack of appropriate support. There is not enough documentation to guide through any bugs or anticipated deployment difficulties. Furthermore, the latest version of the simulator dates back to 2001.

Another simulator worth noting is QualNet [66], a relatively successful commercial simulator that has spawned from GloMoSim. It can be obtained only for a very short evaluation period.

Like ns2, OMNeT++ [22] [63] is an open-source discrete event-driven network simulator. It is also built using C++. OMNet++ is based on hierarchically nested modules. More modules can be created to be used in the simulated environment. Different modules interact by exchanging messages between them. The simulation parameters are set using initialisation script files. OMNet++ also features a rich graphical representation of the simulated network. However, the development of our simple demo on OMNet++ proved to be much more difficult a task than anticipated.

NCTUns [62] is a versatile UNIX-based simulator. The core is open-source; making it easy for any developer to use customised protocols or networking devices by integrating them with the simulation engine using a set of APIs. NCTUns can also be used as an emulator. We have chosen not to use NCTUns for the lack of sufficient support and documentation.

SWANS is a scalable wireless network simulator which runs in a Java environment provided by the JiST platform [6] [61]. The developer claims that it is as flexible as

ns2 with added support for extremely large networks. However, several difficulties were faced in order to get all the required libraries. This has resulted in not being able to get JisT working.

Prowler [65] is a MatLab-based simulation environment. It is well known as a tool for rapid prototyping rather than one used to fully evaluate a protocol. Prowler only supports different parameter settings, but does not support the definition of complex node behaviour.

Table 4.1 summarises the experience of developing a demo using each simulation environment. Different aspects of each simulator in relation to the project are rated, 1 being the highest and 4 the lowest. OPNET, and QualNet were not included as we did not have the chance to test them.

	ns2	GloMoSim	OMNet++	NCTUns	SWANS	Prowler
support for wireless networks	2	1	2	1	1	4
defining and modifying protocols	1	1	3	2	-	4
availability of support	1	4	1	3	2	3
installation and debugging	2	4	2	3	4	1
graphical visualisation	2	3	1	1	-	2

Table 4.1: A Comparison between the Tested Simulation Environments

In light of these rough ratings, it has been decided to use the ns2 package to run our series of MANET simulations.

4.2 ns2 Overview

ns2 is an object-oriented, discrete event-driven network simulator which is written in both C++ and OTcl (Object TCL)♦ [44]. The ns2 architecture supports separating the data and control paths. Traditionally, the control path is implemented using C++, while the data path is managed by OTcl script.ns2 also supports splitting the definition of a single object between C++ and OTcl. Furthermore, objects defined in C++ can be used in OTcl and vice versa.

ns2 has two class hierarchies: the *compiled hierarchy* for the C++ classes, and the *interpreted hierarchy* for the OTcl objects. Any object instantiated in either hierarchy has a parallel object in the other hierarchy.

The OTcl simulation script is mainly used to define the simulated network parameters, such as the properties of nodes, links, and networking devices (such as routers), as

♦ OTcl was created by David Wetherall at MIT as an object-oriented extension to TCL (Tool Command Language), which is a popular open-source scripting language used for rapid prototyping. It was developed by Dr. John Ousterhout at the University of California, Berkeley. In many ways, it is analogous to Perl.

well as the topology, traffic patterns, scheduled events, etc. It is also used to define node movements in wireless scenarios.

The results of a simulation are saved in a *trace* file. This file is generated during runtime and it contains all the details about the communication that occurs in the simulated network. After the simulation is over, the trace file can be used to generate a graphical representation of the simulated network using *nam* (Network ANimator), a visualisation tool contained in the ns2 package. The trace file can also be used to extract several network metrics, such as throughput, delay, and different packet statistics. Such metrics, of course, are important for analysing the performance of the simulated network.

ns2 comes with a large amount of C++ and TCL code which implements common protocols and standards of all layers, such as the Ethernet MAC, IP, OSPF, TCP, UDP, FTP, etc. Furthermore, the ns2 simulator is extensible: customised code can be written (in either C++ or OTcl) to implement experimental protocols.

4.3 Our Agent

We have used C++ code to implement what the ns2 developers call an *Agent*. Agent is an ns2 base class used to execute a custom control path anywhere between the MAC and Application layers. Our agent implements the messaging protocol reviewed in the previous chapter, and it is a subclass of the ns2 'Agent' class. Picturing its position within the layered networking model, it would lie just above the IP layer. Figure 4.1 shows a model of the node as we have developed it. We have employed ns2's implementations of the 802.11 MAC, IPv4, and AODV.

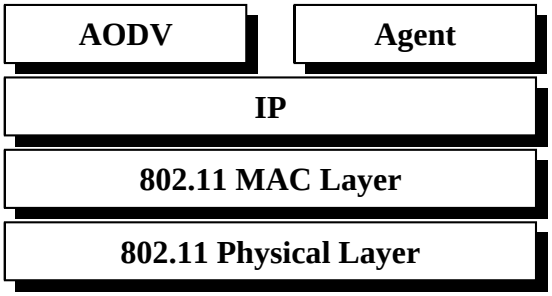


Figure 4.1: Node Model

4.3.1 The Packet Header Definition

In order to implement our agent, we had to define a custom packet which is to be passed between our agent and the network layer where this packet is encapsulated in an IP packet. The fields of our packet are portrayed in Figure 4.2.

Message Type	Catalogue Timestamp	Catalogue / Adverts	Last Received Timestamp

Figure 4.2: Fields of the Custom Header

The "Message Type" field takes one of three values: *Handshake*, *Request*, or *Transfer*. The Handshake message has already been described in section 3.3. The Request message is the message sent by a device once the user is prompted and has picked which new adverts he/she wishes to accept. In a Request message, the third field

contains a list of the identifiers of the requested adverts. The Transfer message is a node's reply to a received Request message. In this case, the third field contains the requested adverts.

4.3.2 Agent Logic & Data Flow

The ns2 'Agent' class contains a public function '`recv(. . )`'. This function is called whenever the Agent receives a packet from the IP layer. We implement our own recv() function, overriding the one in the ns2 'Agent' class, and hence defining the behaviour of our own Agent upon receiving a packet. This, as well as other processing logic included in the file "prj.cc" that implements our agent, is explained below.

I. Receiving a Handshake Message

A Handshake message informs the receiving node of the adverts possessed by the sender. If this message contains a new Catalogue, the receiving node updates it CR table with the latest version of the sender's Catalogue and then updates the user with the new adverts. If the sending node has an old version of the receiver's Catalogue, the receiver responds with a Handshake message of its own.

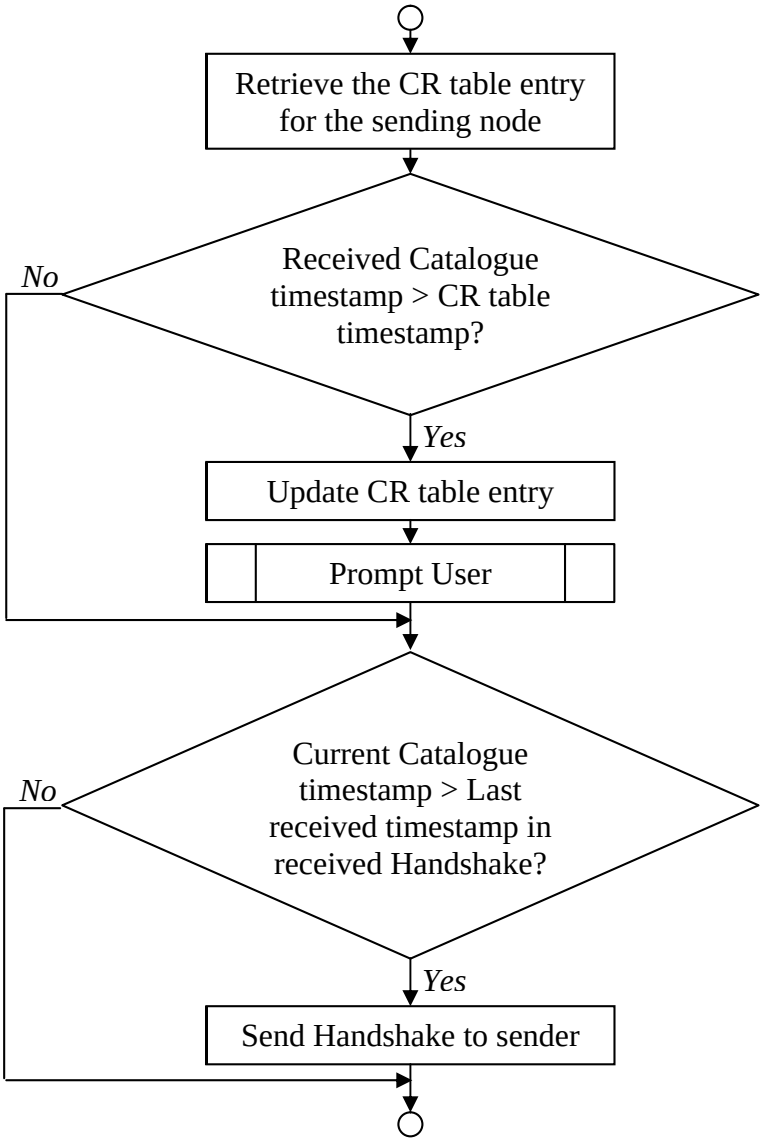


Figure 4.3: Flowchart of Receiving a Handshake Message

II. Receiving a Request Message

A node that receives a Request message will generate a Transfer message containing the requested adverts, and then send it back to the originator of the Request message.

III. Receiving a Transfer Message

Upon receiving a Transfer message, a node first checks its cache for expired adverts. It then stores the received adverts, and updates its Catalogue accordingly. Finally, it updates its neighbours with the adverts it currently possesses, but cancels its next periodic broadcast and reschedules it.

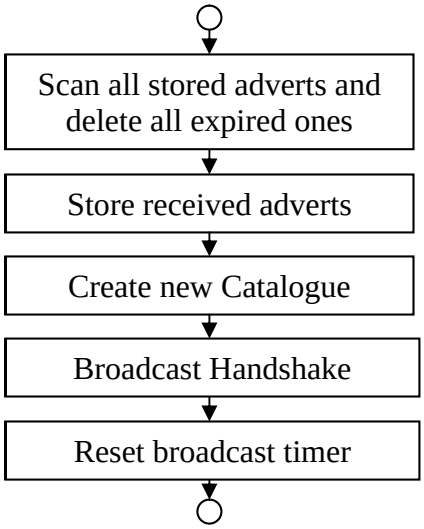


Figure 4.4: Flowchart of Receiving a Transfer Message

IV. Prompting the User to Accept New Adverts

This module uses a random number generator to mimic the user's choice of accepting new adverts. The requested adverts are then listed in a Request message and sent to the node that possesses the new adverts, i.e. the originator of the Handshake message (refer to Figure 4.5).

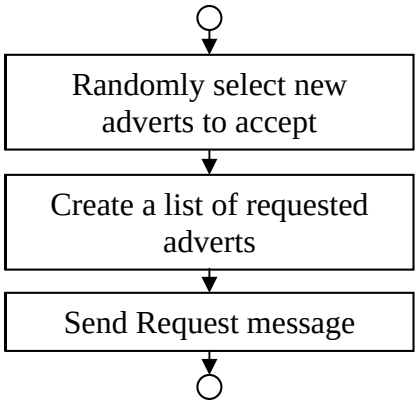


Figure 4.5: Flowchart of Prompting the User to Accept New Adverts

V. Altering the Catalogue Record Table

Whenever an existent Catalogue Record table entry is updated or a new one added, the Catalogue Record table has to be traversed. We use this opportunity to remove any entries older than a particular threshold. This clears the table from entries of "dead" nodes without the expense of periodically scanning it.

4.4 Integrating Our Agent and Packet into ns2

The ns2 package is a vast collection of C++ classes and TCL files with complex interdependencies. Numerous modifications had to be made to the ns2 source files in order to get the simulator to recognise our agent and custom packet defined in "prj.h" and "prj.cc".

4.4.1 Position in the ns2 Class Hierarchy

Since we were implementing our agent in C++, we first had to decide where the agent fits in the ns2 compiled hierarchy. The position of the agent greatly depends on the functionality required from the lower layers. As mentioned in the beginning of section 4.3, our component would depend on the IP layer but not on any transport protocol. Hence, we have decided that our component should inherit directly from the 'Agent' class. Figure 4.6 shows the position of our component in the ns2 compiled hierarchy.

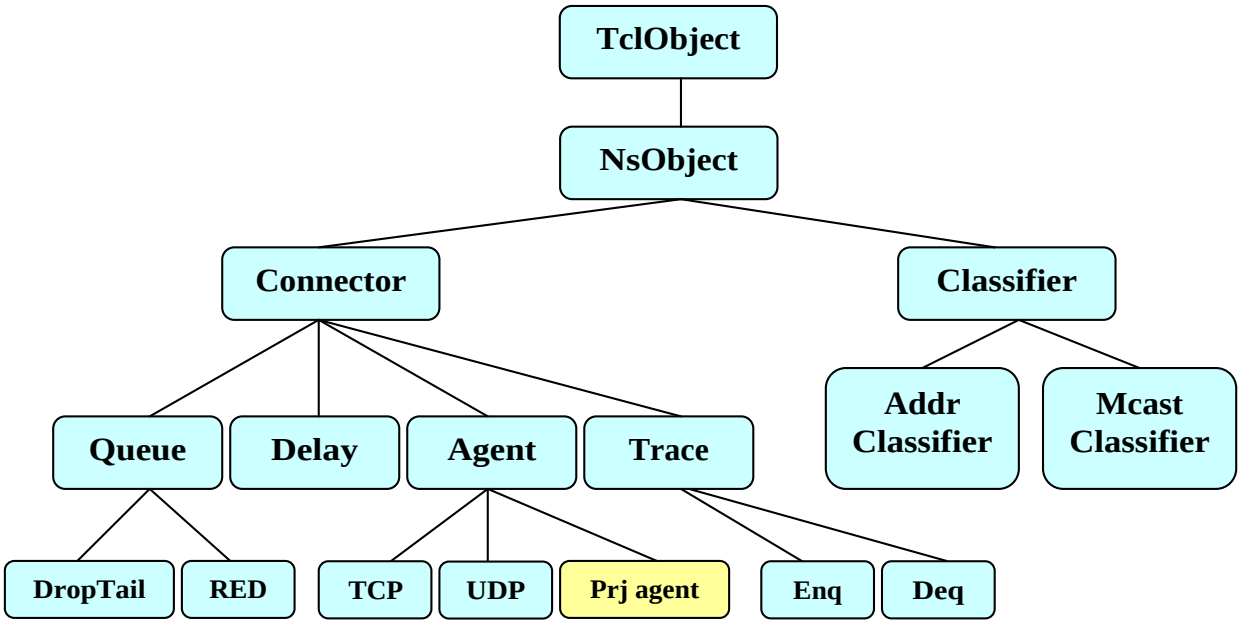


Figure 4.6: The Position of Our Agent in the ns2 Compiled Hierarchy

4.4.2 Introducing Our Packet

Packets are used to exchange data between the different components of ns2. In order to make our packet type usable in ns2, we had to declare its format in a global scope so that all ns2 components would recognise it. This was achieved by adding a few lines to the file common/packet.h, which stores all packet type declarations.

First, we defined a macro to access the field of our packet:

```
63 #define HDR_TCP(p)       (hdr_tcp::access(p))
64 #define HDR_SCTP(p)      (hdr_sctp::access(p))
65 #define HDR_SR(p)        (hdr_sr::access(p))
66 #define HDR_TFRC(p)      (hdr_tfrc::access(p))
67 #define HDR_TORA(p)      (hdr_tora::access(p))
68 #define HDR_IMEP(p)      (hdr_imep::access(p))
69 #define HDR_CDIFF(p)     (hdr_cdiff::access(p))
70 //#define HDR_DIFF(p)      (hdr_diff::access(p))
71 #define HDR_LMS(p)       (hdr_lms::access(p))
72 // added by Yehia
73 #define HDR_PRJ_PKT(p) (hdr_prj::access(p))
74
```

Figure 4.7: The Packet Access Macro in packet.h

Then, we added a definition for the packet type in the declaration of the enumeration `packet_t`:

```
77 enum packet_t {
78     PT_TCP,
79     PT_UDP,
80     PT_CBR,
81     PT_AUDIO,
   ...
172     // added by Yehia
173     PT_PRJ,
174
175     PT_NTYPE // This MUST be the LAST one
176 };
```

Figure 4.8: Packet Type Declaration in packet.h

Next, we specified the packet name:

```
178 class p_info {
179 public:
180     p_info() {
181         name_[PT_TCP]= "tcp";
182         name_[PT_UDP]= "udp";
183         name_[PT_CBR]= "cbr";
   ...
270         // added by Yehia
271         name_[PT_PRJ]="prj";
272
273         name_[PT_NTYPE]= "undefined";
274     }
```

Figure 4.9: Packet Name Declaration in packet.h

Finally, we had to edit two interpreter (OTcl) files. First, we specified any default values of the packet in tcl/lib/ns-default.tcl. In our case, we just added a default packet size of 64 bytes. Second, we stated the packet name for the interpreter to include. This was done by adding the packet name in "`foreach prot {. .}`" in the file tcl/lib/ns-packet.tcl

4.4.3 OTcl Linkage

ns2 offers different ways to link between C++ code and OTcl (interpreter) script. Parameters can be passed between C++ and OTcl. New objects can be created in OTcl based on classes written in C++. This is possible through the class TclClass which provides an interface between C++ and OTcl. Also, script commands can be used to invoke C++ code blocks, and vice versa.

To bind variables declared in C++ from OTcl script, the function `bind()`♦ should be used in the constructor of the C++ class. We have bound two variables: the packet size (if is to be specified) and the packet header offset.

```
28 PrjAgent::PrjAgent() : Agent(PT_PRJ), scan_timer(this) {
29     bind("packetSize_", &size_);
30     bind("off_prj_", &off_prj_);
31
```

Figure 4.10: Binding OTcl Parameters from C++ (prj.cc)

In order to create an instance of our agent for each newly added node, we had to do it from the OTcl script which defines all simulation parameters including nodes. However, we have used the CMU's Pattern Generator tool to create the nodes and their movements for random scenarios, as will be described later in section 5.2. The CMU code supports creating agents and binding them to the created nodes. In spite of this, we did not find the CMU code flexible enough to create script to invoke our agent initialisation and finalisation routines.

To resolve this issue, we settled for an indirect way to initialise the nodes. We modified the supplied AODV code (as well as other ns2 files) so that whenever an AODV agent is created, it also creates an instance of our agent and associates it with the node. The AODV code then calls two functions to initialise our agent instance. When the simulation is over, the AODV code also calls a finalisation function (to display the results).

The first modification was made to the procedure `create-aodv-agent` in the file tcl/lib/ns-lib.tcl. The original and modified versions of the function are respectively shown in figures 4.11 and 4.12.

```
824
825 Simulator instproc create-aodv-agent { node } {
826         #  Create AODV routing agent
827     set ragent [new Agent/AODV [$node node-addr]]
828         $self at 0.0 "$ragent start"     ;# start BEACON/HELLO Messages
829         $node set ragent_ $ragent
830         return $ragent
831 }
832
```

Figure 4.11: The Original create-aodv-agent Procedure in ns-lib.tcl

♦ The function `bind_time()` should be used for time variables, `bind_bw()` for bandwidth, and `bind_bool()` for Booleans. `bind()` is used for variables of any other type.

```
823
824 Simulator instproc create-aodv-agent { node } {
825     # Create AODV routing agent
826     set ragent [new Agent/AODV [$node node-addr]]
827     # create new Prj Agent
828     set prjmgr [new Agent/Prj]
829     # pass the node id to the Agent
830     $prjmgr set-node-id [$node id]
831     # attach the Prj Agent to the AODV Agent
832     $ragent set-prj-mgr $prjmgr
833
834     $self at 0.0 "$ragent start"     ;# start BEACON/HELLO Messages
835     $node set ragent_ $ragent
836     return $ragent
837 }
838
```

Figure 4.12: The create-aodv-agent Procedure after Modification (ns-lib.tcl)

Our agent was then tailored to react to the call "`set-node-id`" (issued in line 830 of ns-lib.tcl) by saving the node ID passed from the simulation script. The node ID will be used to differentiate the nodes in the output of the simulation. Figure 4.13 shows the code for the `command()` function which handles script commands from the interpreter in C++.

```
53 int PrjAgent::command(int argc, const char*const* argv) {
54   if (argc == 3) {
55       if (strcmp(argv[1], "set-node-id") == 0) {
56             node_number = atoi(argv[2]);
57             initiate_adverts();
58             return TCL_OK;
59       }
60   }
61
62   // if the command hasn't been processed by PrjAgent()::command,
63   // call the command() function for the base class
64   return (Agent::command(argc, argv));
65 }
```

Figure 4.13: Interpreting the set-node-id Command (prj.cc)

Similarly, the AODV `command()` function was changed to interpret the OTcl call "`set-prj-mgr`" (lines 106-109 of aodv/aodv.cc). Upon issuing the "`set-prj-mgr`" command, the AODV code creates an instance of our agent, and binds it to the node. It also saves a pointer to the created instance for later reference when the finalisation routine needs to be invoked (lines 139-141 of aodv/aodv.cc). In addition to the mentioned modifications to the aodv.cc file (shown in Figure 4.14), an instance variable `prj_messaging_agent` was added to the file aodv/aodv.h.

It should be noted, however, that these modifications to the AODV code are very slight and should not affect the performance of the simulation. However, the ns2 package is now set up to serve our cause. The ns2 package should be restored with the original files if it is to be used for any other simulation.

```
 78 int
 79 AODV::command(int argc, const char*const* argv) {
    ...
105     // added by Yehia
106     else if (strcmp(argv[1], "set-prj-mgr") == 0) {
107         prj_messaging_agent = (PrjAgent*) TclObject::lookup(argv[2]);
108         return TCL_OK;
109     }
    ...
138     // added by Yehia
139   if (strcmp(argv[1], "reset") == 0) {
140     prj_messaging_agent->display_status();
141   }
```

Figure 4.14: The Modifications to aodv.cc

4.4.4 Adding Our Object File

In the end, we ensured that our agent code is compiled by adding the path of our object file (prj.o) to the list kept in 'Makefile'.

4.4.5 Other Changes

Other ns2 files were modified during implementation, including:

- tcl/lib/ns-agent.tcl: to define the initialisation of our agent; and
- trace/cmu-trace.cc and trace/cmu-trace.h: As mentioned before, ns2 creates a *trace* file which contains detailed data about every transmission event which occurs during the simulation, such as sending, receiving and dropping packets. We have modified the files to render a specific output format for events concerning our packets.
- prj.cc: In our design, we planned to send out broadcast messages with a TTL value of 1 to avoid uncontrollable network flooding. It was later found that the AODV code decrements the TTL value of a packet twice before it is completely handled. Therefore, we changed our code to set the TTL value to 2.

4.5 Concluding Comments

Chapter 4 described the choice of simulator and described our implementation using this simulator. The chapter also shed some light on the lengthy process of integrating the implemented application into the simulation environment.

However, learning to use a complex package such as ns2 in such a short period of time was not an easy thing to do. The learning curve was, and had to be, very steep. The same could probably be said about other simulators; yet from our experience, the ns2 package needs more time to fully understand it. It is the author's fear that this has caused less focus on the optimisation of the application code. Indeed, it was very difficult to implement the Weak DAD addressing scheme.

Furthermore, working with a simulation package has driven the implementation into the simulator's own direction of doing things. For instance, ns2 does not allow passing dynamic structures in packets. We have used one packet type to pass 3 different types of control messages. This implementation is, of course, inferior to one where there are three distinct control messages. However, this was not easy to achieve using ns2.

This, of course, was anticipated but not to this extent. The application was stipulated to the simulator's capabilities and limitations, which resulted in an implementation weaker than initially intended. Nevertheless, evaluation of this implementation still stands as a measure of the messaging protocol efficiency and capability to adapt to the circumstances of the target scenario.

5 Evaluation

We aim at assessing the performance of the solution proposed by this project in the objective scenario. However, implementation of the Weak DAD addressing scheme was not carried out. In this chapter, therefore, we shall present the evaluation of the combination of AODV and our messaging protocol in executing the target application of message propagation over WiFi networks.

5.1 Evaluation Technique

In analyzing the performance of our proposed solution, we have considered four typical performance measures for ad hoc networks: data throughput, end-to-end delay, routing overhead and packet error ratio. In addition, we have measured the amount of broadcast traffic in the network.

- Data Throughput: We have considered the minimum, maximum, and average data throughput of all nodes. Only data packets were considered in calculating this metric, and so the effect of induced routing overhead was avoided.
- End-to-End Delay: This is the average delay from the time that a packet is sent up to the time it arrives at its destination. Many factors contribute to this delay, such as buffering, MAC layer processing, and the physical propagation delay.
- Routing Overhead: This metric was used to evaluate the efficiency of the routing protocol by means of the added overhead. The metric value is equal to the ratio (in percentage) of the total routing Bytes sent to the total data Bytes sent.
- Packet Error Ratio: This is equal to the ratio (also in percentage) of the number of unsuccessfully received packets to the number of packets sent. In wireless networks, packets may fail to arrive at their destination for several reasons. The packet error ratio, sometimes referred to as PER, is an important measure of the behaviour of the network [20]. Hence, we consider it to establish an understanding of how the quality of packet delivery changes using our solution.
- Broadcast Control Traffic: We have tried to study how the minimum, maximum, and average amounts of broadcasted control packets (broadcast Handshakes) change under different network conditions.

The metrics were collected from a number of different simulations, each resembling different network conditions. The aim was to observe the performance of our proposed solution under various configurations of the target network. These configurations were created using different values for the following parameters.

- Network Scale: As scalability is an imperative issue of concern, we have repeated the simulation with different numbers of nodes.
- Node Mobility: We have created several different patterns of node movement in order to investigate the performance in different topology natures, ranging from

stable to highly dynamic. The patterns are differentiated using two parameters: the maximum speed of node movements, and the average pause time between the movements.

5.2 Simulation Models

We have created different scenarios, each with different values of network size, maximum node speed, and average node pause time. All other parameters were kept constant for the simulations. In this section, we will first describe the set of parameters which were fixed throughout the simulations and then introduce the different scenarios of varying parameters.

Varying parameters, on the other hand, were created separately using the CMU (Carnegie Mellon University) Pattern Generator [38]. This is one of the independent tools supplied with the ns2 package. It allows creating random traffic and node movement patterns. We have used CMU's "*setdest*" code to build our scenarios with different values of number of nodes, node speed, and pause time.

5.2.1 Fixed Parameters

Following is a set of network parameters that were kept unchanged for all simulations.

I. Physical Interface Properties

In the OTcl script file "sim.tcl", we have used the default physical interface properties. These properties equip each node in the simulation with a 914MHz Lucent WaveLAN DSSS wireless network interface 150 centimeters above ground level. Such adaptor has a transmission range of approximately 250 meters. The interface queue can hold up to 50 packets.

II. Size of the Topology Ground

The simulated network nodes roam around an area of 500 by 500 meters. This is almost double the diameter of the transmission range. This allows having nodes both in and out of range, given that the CMU code generates random initial node positions.

III. Simulation Time

We have allowed each simulation to run for a period of 5 minutes. This time period was found to be more than enough to circulate all adverts to all nodes in the mentioned topology ground.

IV. Initial Adverts

We have set the maximum number of distinct propagated adverts at 10. This value is appropriate for a simulation period of 5 minutes and in a topology of 500 by 500 meters.

We have created a function which initiates each agent with a random number of adverts, arbitrarily selected from the set of 10 distinct adverts. The function is called every time an agent is created. Hence, each agent starts the simulation with a different set ranging in size between 0 and 10 adverts. This was done to mimic the initial transmission of adverts from an access point to mobile users.

Moreover, each advert in the initial set that a node receives is assigned a random expiry time. This is more realistic than setting an individual expiry time for each distinct advert, because it is anticipated that the provider would disperse adverts with different expiry times from different access points according to marketing strategies.

V. Broadcasting Period & Threshold

Nodes were set up to broadcast every 40 seconds. This should be a long enough period to reduce both the induced control traffic and the periodic processing. Catalogue Record table entries older than a threshold of 60 seconds are deleted. This is well over the broadcast period plus the average propagation delay.

VI. Node Joining Time

Nodes were set up to start broadcasting, hence declaring their addition to the network, at random times after the first second of the simulation. In a real network, nodes are not expected to start broadcasting all at once.

5.2.2 Varying Parameters

In the scenarios we created, we have changed one parameter at a time while keeping the others constant to investigate the effect of only one varying parameter at a time. However, the values of the non-changing parameters were set in order to mimic the target network to the best of our knowledge.

I. Number of Nodes

We have used 8 different values of number of nodes: 10, 15, 20, 30, 40, 50, 70, and 100. When the network size was kept steady, we used 20 nodes in our network simulation. We believe that this is a reasonable network size that allows the observation of altering the mobility of the nodes in a 500 by 500 meters topology.

II. Maximum Node Speed

The 8 different values for this parameter were 0.01, 0.5, 1.0, 1.5, 2.0, 3.0, 4.0, and 6.0 meters per second. The nature of the simulated nodes using these values ranges between stationary[♦] and highly mobile. We have selected the value of 1 meter per second as a default value for the maximum node speed when the other parameters are being changed. Nodes moving at a maximum of this speed would create a dynamic MANET, which is where we would want to deploy our application.

III. Average Node Pause Time

Again we have used 8 different values: 0, 1, 2, 4, 8, 20, 40, and 100 seconds. The default value was 2 seconds in order to create a continuously changing topology where the nodes tend to move all the time.

♦ Since the CMU Pattern Generator does not allow a maximum speed of 0, we used 0.01 instead to create a scenario with stationary nodes.

5.3 Simulation Runs

Different scenarios generated by CMU code were saved in separate files. Fixed parameters were defined in the C++ header file "prj.h" and the OTcl simulation script file "sim.tcl". After integrating our agent and recompiling ns2, the sim.tcl file is used as input to the simulator. The simulation script then loads one scenario at a time, and outputs a text-based packet trace file (sim.tr), as well as another file used for the visualisation of the network (sim.nam).

We have written an AWK [46] script file to process the "sim.tr" file and compute the metrics we need for the evaluation. The metric values from the 24 different scenarios were used to plot the graphs discussed in the following sections.

All tests were carried out on an Intel Celeron 2.00 GHz processor, with 1 GB of RAM. Since ns2 only runs on UNIX, Cygwin [43] was used to create a Linux-like environment on top of Windows XP. A few of the tests were also carried out on Linux Ubuntu release 6. Once the results from the Linux simulations were found to be identical to the ones from the simulations run on Windows, we continued to run the rest of the simulations using Cygwin over Windows XP for convenience.

5.4 Results & Quantitative Analysis

This section exhibits and discusses the gathered metrics. Please refer to the website for more detailed simulation results.

When one of the parameters is changed, the default values of the other two are maintained as mentioned in section 5.2.2 and restated in Table 5.1 below for clarity:

Parameter	Number of Nodes	Maximum Speed	Average Pause Time
Default Value	20	1 meter/second	2 seconds

Table 5.1: Default Evaluation Parameter Values

5.4.1 Broadcast Control Traffic

We have chosen to discuss the value obtained for this metric first as it was found to have a profound effect on the values of the other metrics. Figures 5.1 through 5.3 show the maximum, minimum, and average numbers of broadcasted Handshake messages sent per second.

This metric is very important to estimate the amount of induced network overload. It also gives us an indication of how much processing do we induce when the nodes receive broadcasts. We measured the outgoing broadcast messages rather than the incoming ones for two reasons:

- Sent messages are not necessarily received if nodes are out of range, yet they still occupy bandwidth; and
- A single sent broadcast message is received multiple times. Using the number of received messages would require normalisation with respect to the number of receiving nodes, which is more complex to calculate.

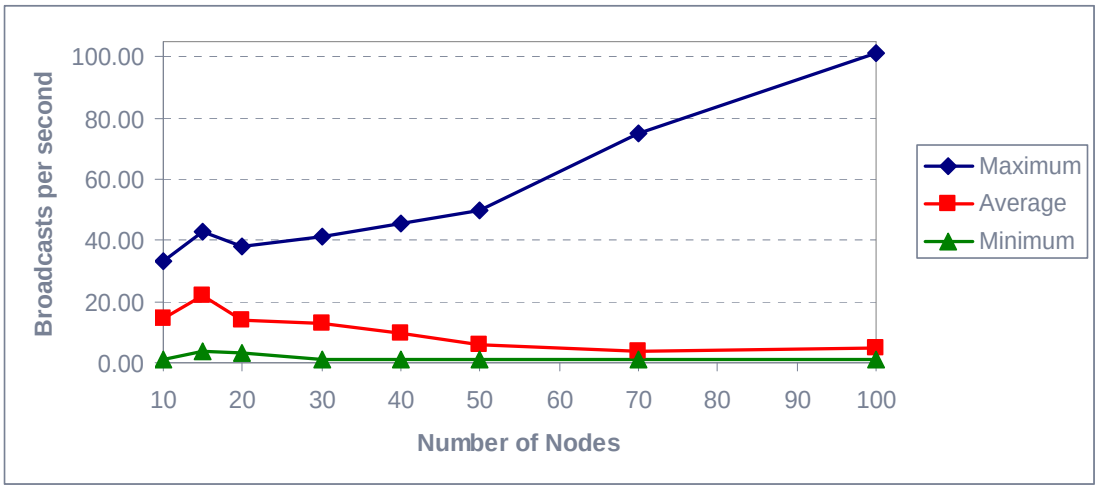


Figure 5.1: A Plot of Broadcasts per Second vs. Number of Network Nodes

Examining the amount of broadcast control messages as the network scaled brought on a very interesting deduction: The average number of broadcasted Handshake messages decreases as more nodes join the network. While the average value is 14.63 per second for a network of 10 nodes, it decreases to 3.69 per second for 70 nodes and 5.06 for 100 nodes. Considering that the broadcast timeout period is 40 seconds, for the 100 node network, the average broadcast per timeout period per node is 2.024 and the minimum is 0.400. The maximum, however, has an almost linear relationship with the number of nodes.

This can be easily explained. Nodes newly joining the network are triggered to broadcast by default, hence the (almost) linear increase in the maximum number of broadcasts per second. However, all nodes that receive a broadcast would not reply with a broadcast message, but instead with a unicast message, if any. Moreover, as nodes start to receive adverts from each other, they reschedule their timers resulting in a reduced magnitude of broadcast traffic.

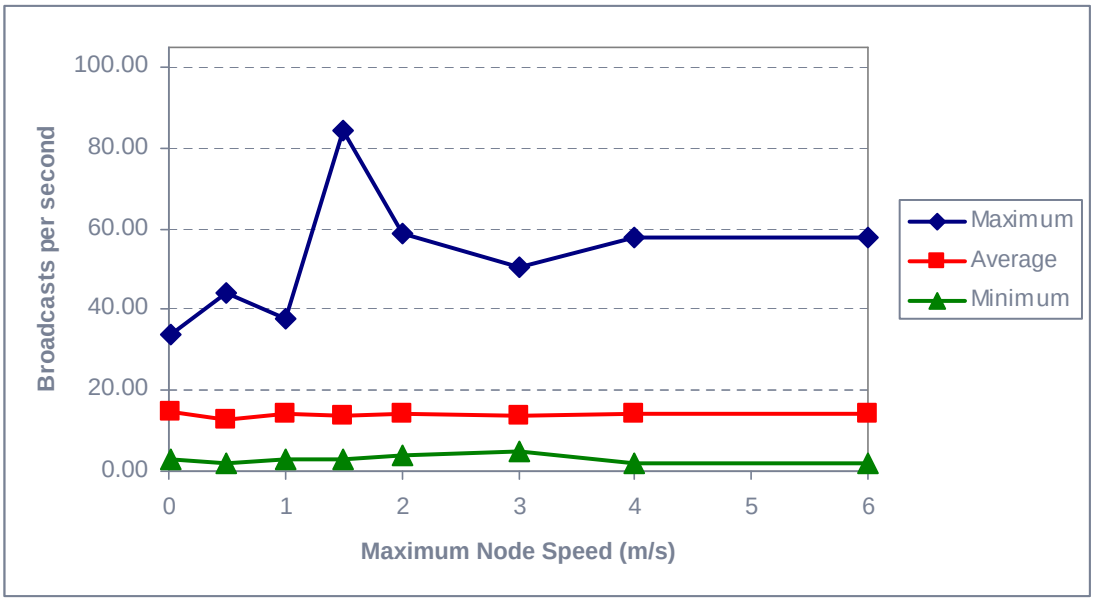


Figure 5.2: A Plot of Broadcasts per Second vs. Maximum Node Speed

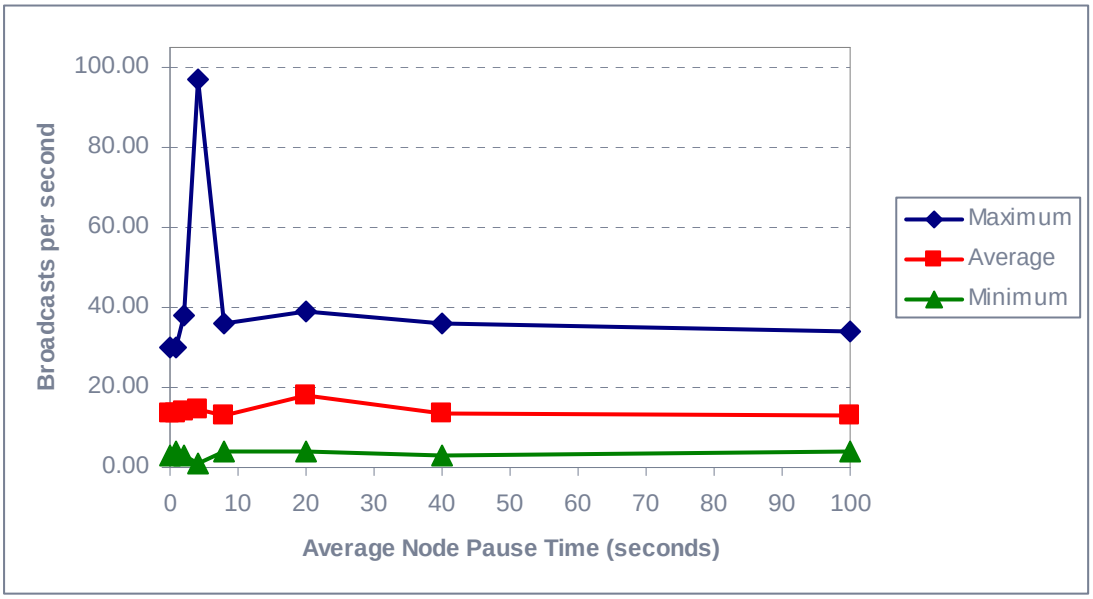


Figure 5.3: A Plot of Broadcasts per Second vs. Average Node Pause Time

The change in the size of broadcast traffic does not seem to have a clear relationship with the change in node mobility. The average and minimum number of broadcast messages per second is almost steady around 14 and 3, respectively, with a standard deviation less than 1.5. Slight fluctuations in the average and minimum values, as well as peaks in the maximum values, can be attributed to the difference in node addition times in the different scenarios.

5.4.2 Data Throughput

The throughput was calculated by measuring all data Bytes received by the nodes. From these measurements, the minimum, maximum, and average data throughput values were used to plot the following graphs (figures 5.4 – 5.6).

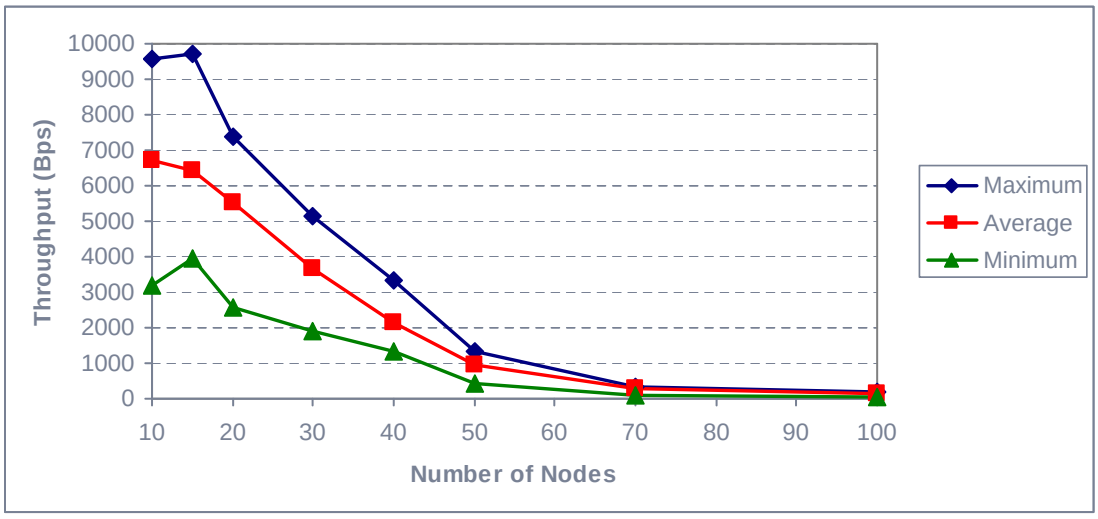


Figure 5.4: A Plot of Data Throughput vs. Number of Network Nodes

The throughput decreases as more nodes are included in the network. This is due to the significant decrease in the number of broadcast messages, mentioned in the previous subsection.

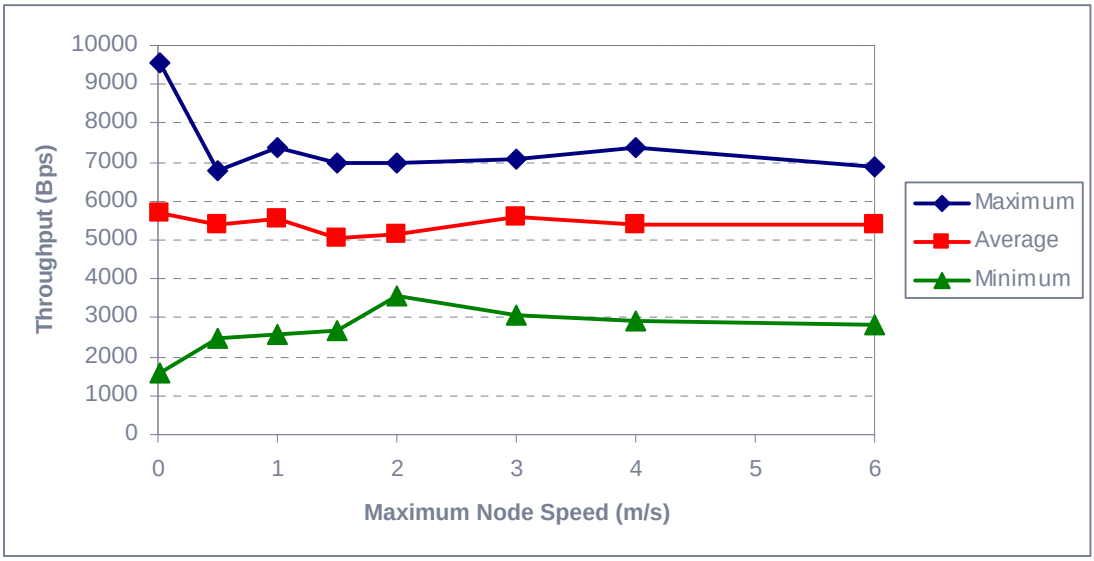


Figure 5.5: A Plot of Data Throughput vs. Maximum Node Speed

In wireless communication, the closer the directly communicating peers are to each other, the less the possibility of interference, and the less the probability of transmission error. In other words, the closer the peers are to each other, the more the probability of successful transmission and hence higher throughput.

Stationary nodes have fixed distances between them, and hence produce both the highest maximum and the lowest minimum throughput values. As the nodes tend to move, the mean inter-node distance does not change much (and neither does the average throughput), but the standard deviation of the inter-node distance decreases. This explains the decrease in maximum throughput and the increase in minimum throughput.

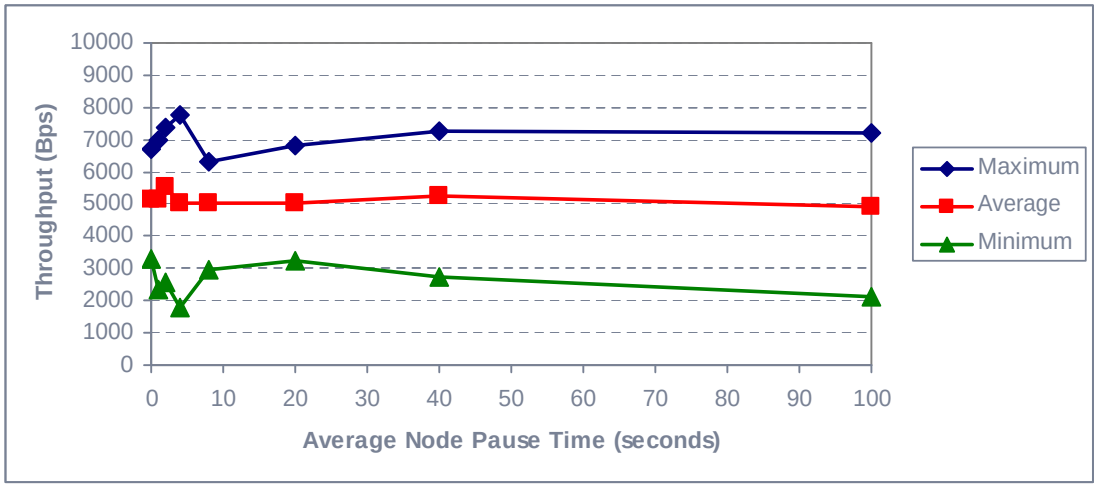


Figure 5.6: A Plot of Data Throughput vs. Average Node Pause Time

The test with the zero average pause time represents a network where nodes are constantly moving. This continuous movement has resulted in a maximum throughput 5% lower than the average obtained using the different pause times. Average pause times between 1 and 4 seconds have resulted in the highest maximum throughput levels (about 10% higher than the average maximum throughput). However, they also

produced the lowest minimum throughput. It is clear how low node pause time causes nodes to move in and out of the range of other nodes at a high rate. This causes the peak in the maximum throughput as well as the dip in the minimum throughput for low node pause times.

By looking at the average data throughput values in figures 5.5 and 5.6, it is clear how AODV copes with changes in node mobility to support a steady average throughput.

5.4.3 End-to-End Delay

The minimum end-to-end delay was 0.0018 seconds for all simulations. This is equal to the average propagation delay. The maximum values were abnormally high due to a limited number of packets which suffered from high delay due to buffering and/or edge-of-range transmission. Thus, we excluded the minimum and maximum and decided to focus on the average delay.

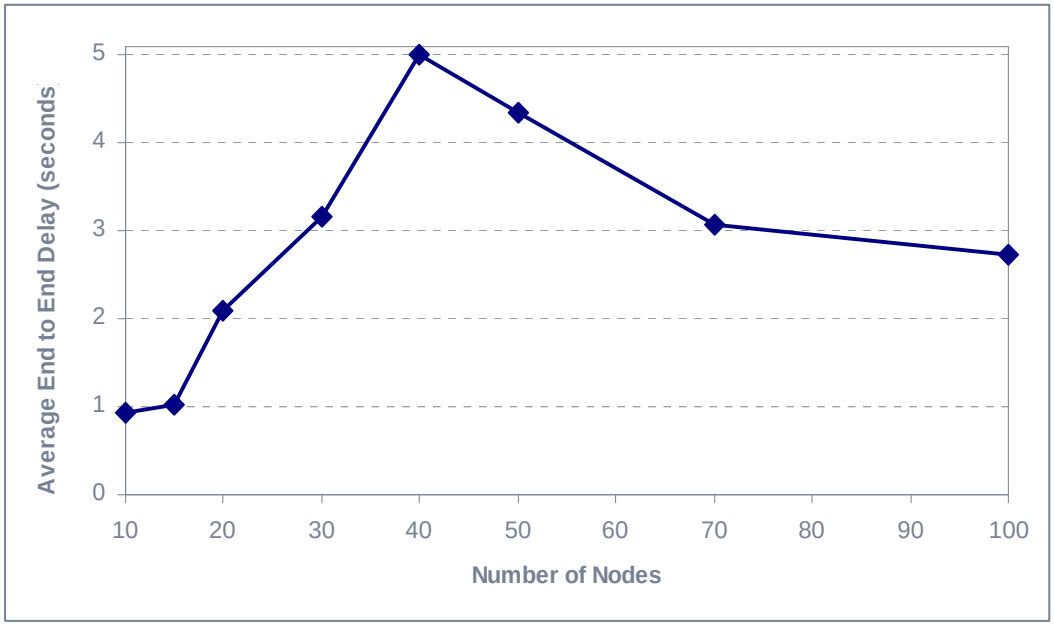


Figure 5.7: A Plot of Average End-to-End Delay vs. Number of Network Nodes

We can see from Figure 5.7 that, as the network grows, the average end-to-end delay increases steeply reaching a peak of 5.0125 seconds. The average end-to-end delay then begins to dip steadily to reach a value of 3.0646 seconds, and then further decreases as the number of nodes in the network increases from 70 to 100, eventually reaching a value of 2.7278 seconds. It is unclear why the average end-to-end delay shows such changes.

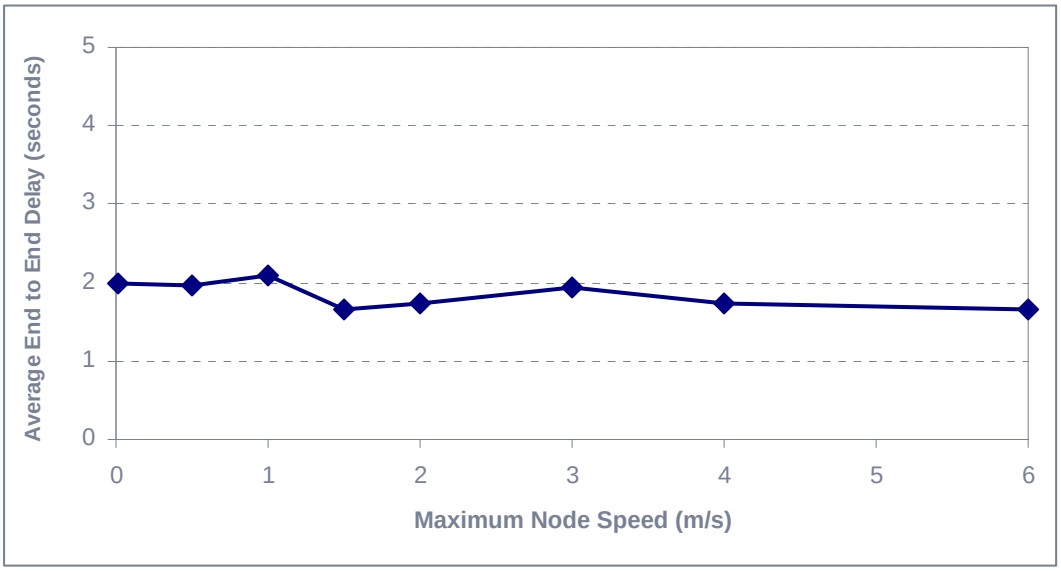


Figure 5.8: A Plot of Average End-to-End Delay vs. Maximum Node Speed

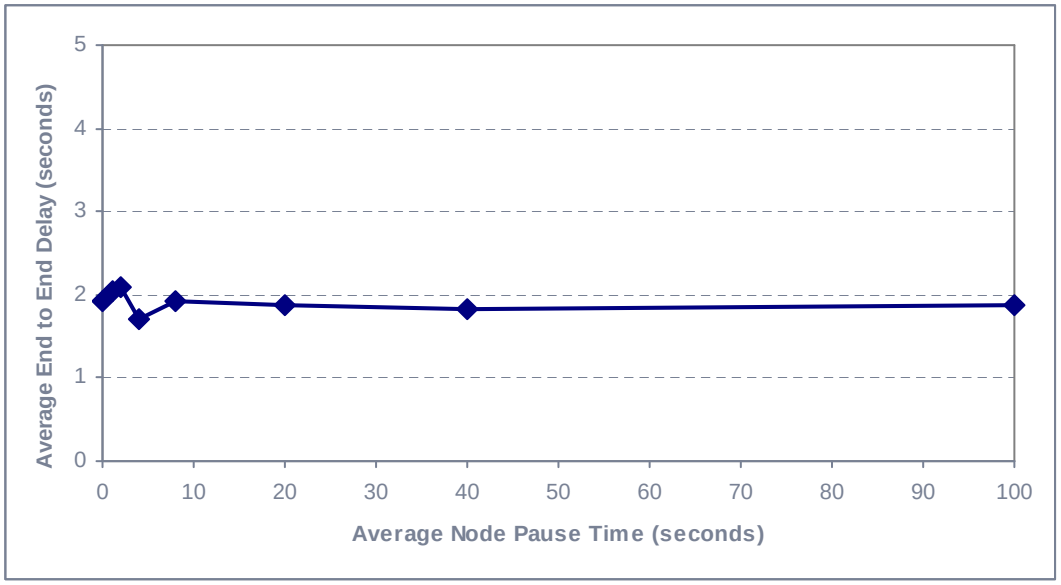


Figure 5.9: A Plot of Average End-to-End Delay vs. Average Node Pause Time

As figures 5.8 and 5.9 show, the average end-to-end delay shows little change to node mobility. This demonstrates the adaptability of AODV to the change in node mobility, continuing to provide a steady average delay around the value of 2 seconds.

5.4.4 Routing Overhead

The routing overhead is equal to the ratio between the total routing packets sent and the total data packets sent.

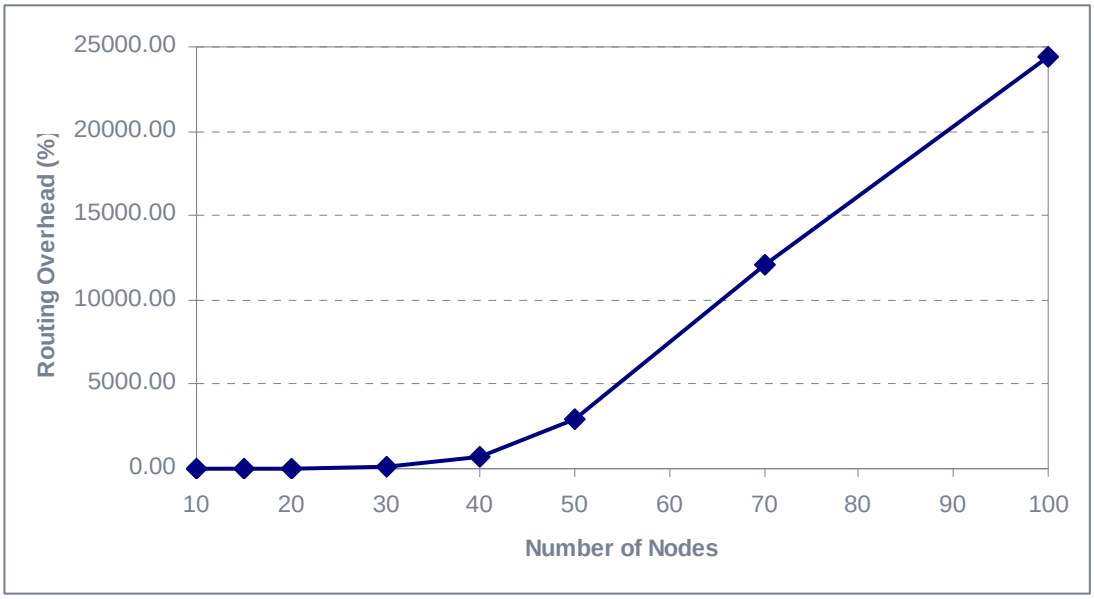


Figure 5.10: A Plot of Routing Overhead vs. Number of Network Nodes

Figure 5.10 shows a sharp increase in the routing overhead caused by the addition of nodes to the network. Such unusual rise may be partly due to the diminishing amount of broadcast messages, which constitute the most part of the denominator in the formula for the routing overhead, as the network grows. However, the numbers are alarming.

Therefore, we looked deeper into the AODV code used. It was found that routes older than 10 seconds are deleted. We experimented with this factor and concluded that the overhead decreases to around 19,900 % (instead of the earlier value of 25,000 %) when active routes are maintained for 100 seconds. This is only a slight improvement, but it shows that the decline in broadcasts causes routing table entries to expire and hence triggers more route requests. Increasing the lifetime of the routing table entries decreases the routing overhead for larger networks.

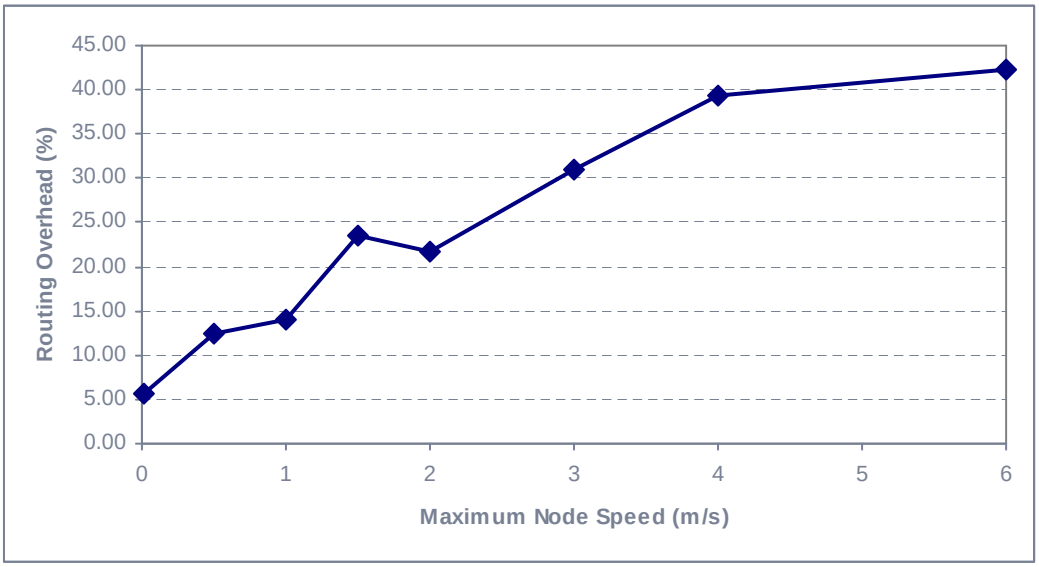


Figure 5.11: A Plot of Routing Overhead vs. Maximum Node Speed

We can identify increase in node mobility by an increase in node speed, a decrease in node pause time, or both. Regardless of the cause, high node mobility leads to link failure and consequently triggers AODV to send route request packets. This relationship is apparent from Figures 5.11 and Figure 5.12.

Reading into the graph shown in Figure 5.11, the routing overhead increases from 5.55% in a network with almost stationary nodes to 30.95% in a network whose nodes move at a maximum of 3 meters per second. This is a relatively subtle increase in routing overhead for such a significant increase in node speed.

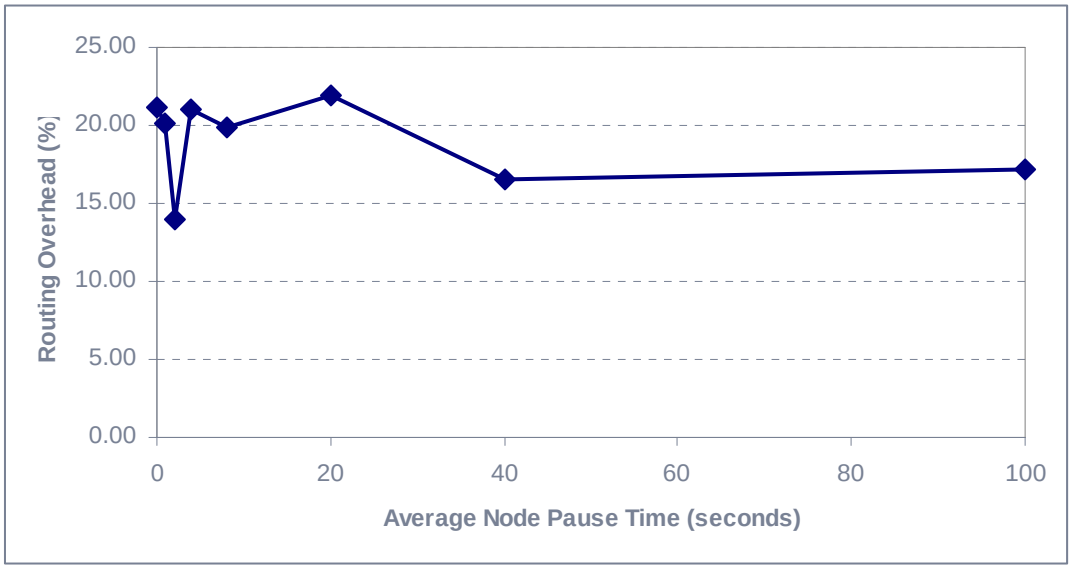


Figure 5.12: A Plot of Routing Overhead vs. Average Node Pause Time

5.4.5 Packet Error Ratio

This packet error ratio is equal to the ratio of the number of received packets to that of the ones sent. Since our protocol utilises broadcast traffic, a single packet sent may be received several times and hence the packet error ratio is sometimes negative.

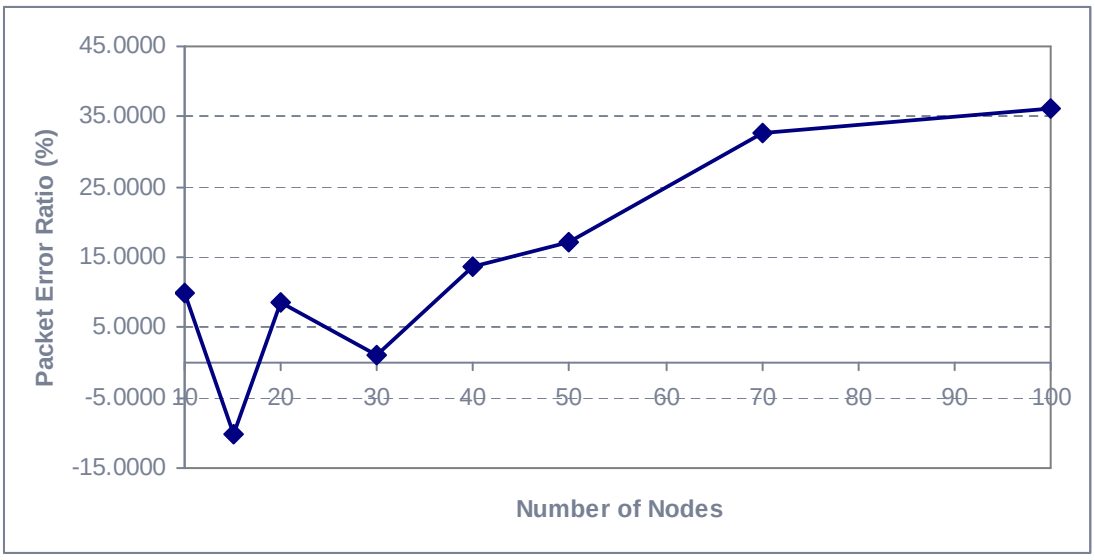


Figure 5.13: A Plot of Packet Error Ratio vs. Number of Network Nodes

The packet error ratio increases as the network grows. This is mainly due to packet collisions which result from high routing overhead in large networks, as mentioned in section 5.4.4.

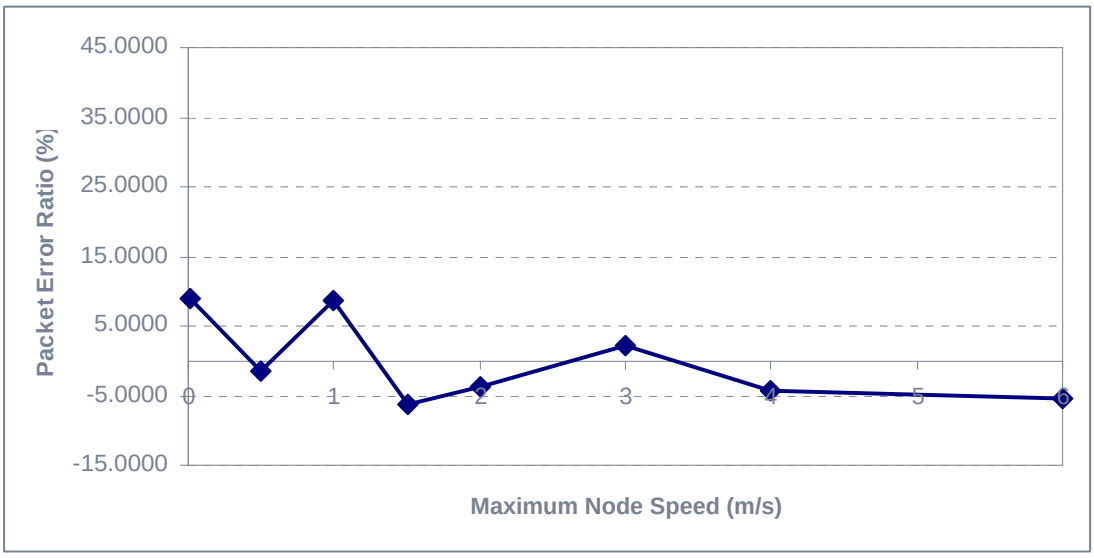


Figure 5.14: A Plot of Packet Error Ratio vs. Maximum Node Speed

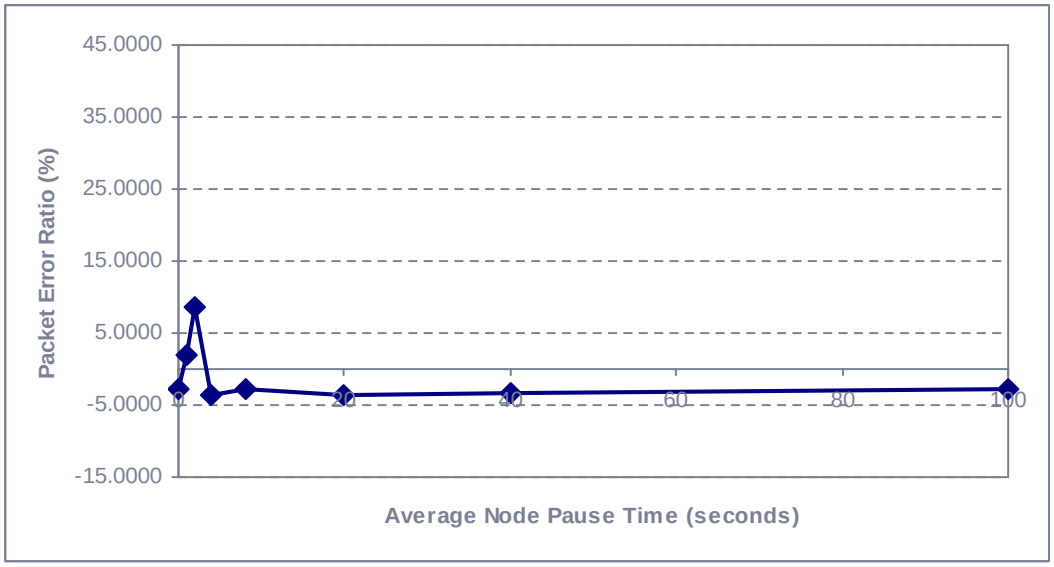


Figure 5.15: A Plot of Packet Error Ratio vs. Average Node Pause Time

The last two figures portray how AODV maintains a relatively low packet error ratio regardless of the mobility of the nodes.

5.5 Qualitative Analysis

5.5.1 Observations

AODV has proved to perform well under conditions of high node mobility. The average end-to-end delay, average throughput, and packet error ratio levels showed very weak relationships to change in node mobility. Because AODV is an on-demand routing protocol, it retrieves fresh routes providing steady average values of

throughput, end-to-end delay, and packet delivery ratio. This adaptation to dynamic topologies, however, comes at a cost of increased routing overhead.

While the messaging protocol used a low magnitude of broadcast traffic in large networks, the routing protocol did not react so gracefully to increasing the number of nodes in the network. The routing overhead increased exponentially. The average end-to-end delay was less dramatic, yet more confusing. The delay increased linearly with growing network size to reach just above 5 seconds for 40 nodes. The delay then improved for larger networks, averaging less than 3 seconds for a 100 node network.

5.5.2 Application and Simulation Model Parameters

The implementation and simulation models had many fixed parameters, such as the broadcasting interval, message size, the wireless interface length, etc. More tests can be made using different values of these fixed parameters. Such tests aim at optimising the messaging protocol, defining the optimal message length, and/or providing a more realistic simulation of the network. For example, instead of changing the AODV code to keep routing table entries for a longer period of time, the broadcasting timeout period could be changed to reach a point of balance between routing overhead, network overhead, and node processing.

However, we have concentrated our efforts on testing three main parameters that represent network scale and node mobility. The results obtained, therefore, provide an insight to the performance of the solution under different conditions of network size and topology dynamics.

The simulation model could be replaced with one which uses a different generator for the node movement patterns. This is one of the setbacks of using simulators, as we elaborate in the following subsection.

5.5.3 Use of a Simulated Environment

A network simulator synthesises the conditions of a real network in order to use it for experimentation at a low cost. Nevertheless, this created environment is based on a certain understanding of the real environment that it is supposed to mirror. Discrepancies arise from either poor understanding of the real network properties, or from implementation limitations. One discrepancy that has been mentioned is the node movement patterns. Others include radio interference, obstacles, and unpredictable user behaviour. Hence, simulated networks can never recreate real networks completely. With that in mind, we have tried to make the simulation as close as possible to the characteristics of the real network.

6 Conclusion

In this chapter we draw some conclusions about the work carried out in this project. We review the main contributions of the research, and discuss directions for future research.

6.1 Conclusions

This report proposes an implementation which enables exchange of content in a network that can be set up spontaneously. The network does not need any infrastructure to be established ahead of time; it can be set up anywhere and at any time. On top of the flexibility this solution offers, it can outperform traditional wireless networks that are set up using an access point. Unlike in an infrastructure-based wireless network, the average bandwidth per mobile device in an ad hoc network does not deteriorate as the network grows.

The project includes substantial research in the context of the target network. The project stresses on three main aspects; namely addressing, routing, and message exchange.

6.1.1 Addressing

After thorough investigation of the network properties, Weak DAD was proposed as the addressing scheme that best suits the target network. The Weak DAD scheme provides an effective and lightweight mechanism for nodes to acquire an IP address in a MANET. However, it was not possible to test the Weak DAD addressing scheme within the time of the project, and consequently, it was not evaluated within our target scenario.

6.1.2 Routing

An extended effort was made to choose a routing protocol that is suitable for the circumstances of the target network. A vast number of protocols were taken into account by categorising them. The effort concluded in selecting AODV. The results have shown that AODV performs well under high node mobility at the cost of added routing overhead.

6.1.3 Message Exchange Protocol

The project included the design, implementation, and evaluation of a novel protocol that handles the propagation of content (i.e. adverts) across the network. The simulations demonstrated the efficiency of the protocol in exchanging content using short control messages. The small size of the control messages helps in reducing the probability of collisions. Moreover, the number of broadcasted control messages decrease for large networks, reducing collisions and improving packet delivery ratio.

Although we have tried to reduce the processing made by the application by not choosing to scan the Catalogue Record table periodically, the receipt of periodic broadcast updates forced nodes to check their CR tables anyway (to retrieve the most recent timestamp). An improvement could be to keep a cache of the most recently used Catalogue Record table entries. However, this might still not serve as being the solution as nodes who have recently broadcasted will probably not broadcast again until their timer expires, unless of course they update their Catalogue. Yet, this is an approach worth investigating in future work.

The protocol runs over IP and AODV, although AODV can be replaced with another routing protocol. The protocol does not rely on any change to the layers below and including the Network layer. This facilitates easy and rapid deployment on real devices.

The assumptions that the messaging protocol was based upon have all been solid. For example, real deployment of the protocol requires tight synchronisation of the clocks of all nodes. This assumption is valid as the 802.11 MAC layer provides clock synchronisation, as explained in section 4 of Appendix B.

However, the process of implementing the protocol was a lengthy one. Despite the flexibility that ns2 offers, learning to use such a complex package in the time frame of the project was not an easy task. The learning curve was, and had to be, very steep. It is speculated that the same would have been experienced if any other simulation environment had been used instead of ns2, although this might not necessarily be true. It is the author's fear that the difficulty entailed by using ns2 has caused less focus on the optimisation of code for the message exchange protocol.

6.1.4 The Evaluation Process

The evaluation of the solution adopted several different scenarios with parameters which were not completely realistic (such as high node speeds), but in effect, they served in examining the message exchange protocol and the routing protocol under severe conditions of the target network.

The evaluation revealed the success of AODV and the messaging protocol in handling dynamic networks, but they also revealed the flaw of extremely high routing overhead in large networks. This is a field for future work.

6.2 Contributions

This project lays groundwork for the application proposed by BT, as well as potential applications over mobile ad hoc networks. The research tackles the major networking issues related to such applications. Thorough and careful analysis of the addressing and routing concerns has been carried out. The application presented in this report, unlike many others, does not require each node to have a predefined address. Instead, we assign addresses as nodes join the network. The application also induces relatively low routing traffic.

The work also includes the design and implementation of a lightweight message exchange protocol. The protocol is suitable for mobile ad hoc networks populated by laptops, personal digital assistants (PDAs), or WiFi-equipped mobile phones. Using

timestamps, the mutual update between nodes does not have to be two-way all the time, thus reducing the overhead imposed on the network. This not only saves network bandwidth, but also reduces the probability of collisions, hence yielding improved communication.

6.3 Future Work

6.3.1 Addressing

As Weak DAD requires close integration with routing, it should be tested within the restrictions of the target network after integration with AODV. In order to properly evaluate the scheme, the tests should measure several metrics such as the minimum and maximum delay for a node to acquire an address, and the network overhead induced by an introduction of *n* nodes (where n is a variable between 1 and an assumption of the maximum number of nodes in one network).

Future work may also focus on other addressing schemes. However, if a timeout-based scheme, such as DAD, is chosen, then the process of setting the timeout period should be investigated to reach a description of the optimal timeout period and how this period changes as other parameters, such as network size, change. This is an important issue that directly affects the delay before acquiring an address.

6.3.2 Routing

After taking all classifications of ad hoc routing protocols into account, a certain set of protocols were chosen as contenders to operate in our target network. This set included protocols which are reactive, location-unaware, and flat. From this set, we have picked DSR and AODV since they are the only de facto standard routing protocols. Future work can focus on implementing other protocols from that set and comparing them against AODV.

6.3.3 Complete Solution

Future work should work on building the complete application, using the outcomes of both this project and the other BT project which complements this one. In such a study, results from both projects should be integrated to provide a complete solution. The study should look into the consistency of the results of both projects, and whether or not there are any cross-layer discrepancies. For example, the protocol proposed in this project ran just above the IP layer. Future work may consider building the application above UDP or TCP if this provides more support to the upper layers (e.g. security), making sure that this would cause no problems with AODV.

Once the whole application is ready for deployment, some work may be done on more integration between AODV and the messaging protocol. For example, updating the Catalogue Record table every time the AODV routing table is updated could be studied. The performance of the application after such integration should be evaluated. Nevertheless, it is to be noted that more integration with AODV would necessitate further work if testing with other routing protocols is desired.

6.3.4 Better Simulations

More complex simulation tests can be carried out to investigate the effect of more protocol and network parameters, as discussed in section 5.5.2.

6.3.5 Testing in Real Networks

The simulation of a network is merely a representation of its real counterpart. Such representation is based on properties and measurements of the real network environment as perceived by the developers of the simulation environment. These measurements can never be accurate enough so as to create an exact duplicate of the real network. Hence, results obtained from simulated networks can be, and to a great extent, inaccurate leading to erroneous conclusions.

Future studies need to test the proposed application in a real environment, where many factors that are not considered in simulation can affect the performance. These factors include radio interference, collisions, and obstacles among others. Such factors are totally unpredictable in a real environment and hence can never be properly represented in a simulation. Moreover, there are certain difficulties that can be faced when deploying in a real environment, the most common of which is of course the problems associated with working in kernel space. While debugging using simulators in user space is straightforward and platform-independent, debugging in kernel space is difficult and time-consuming as it is customarily not fully supported by the operating system. Deploying on real devices is also significantly platform-dependant. Such difficulties are never foreseen until stumbled upon, and in many cases can affect the design of the solution.

It is, however, infeasible to build large test-beds. In this case, it is suggested to deploy the solution on as many nodes as possible. If testing in real networks is not at all feasible, future studies can work on making the simulation closer to reality by incorporating as much as possible of the aforesaid factors of real networks. This latter approach, however, is not recommended.

6.3.6 Multiple Addresses

In this project, we have assumed that all devices own a single address, if any. This is a fair assumption, since all current mobile devices contain a maximum of one WiFi adaptor. If, however, the devices had the need to acquire more than one address, then we would need to repeat our search for suitable addressing and routing protocols. The AODV protocol can only identify a node by a single IP address and is hence not suitable for a network where nodes can have more than one address. Similarly, Weak DAD should be reconsidered.

6.3.7 Multicasting

This project has focused on unicast content forwarding, although content was usually transmitted to more than one recipient. Future work can question the possibility of using multicast groups in forwarding the content across the network. The primary concern regarding multicasting is whether or not it would be feasible to maintain multicast groups in unstable networks such as MANETs.

6.3.8 Quality of Service

Analysis of the importance of using QoS in the target network can be the topic of future projects. QoS provisioning is not an easy task, even within wired networks. However, the introduction of QoS provisioning in MANETs would promise a wide range of new and more advanced applications. For one, QoS would allow the implementation of applications that rely on real time traffic. There is great potential in this area, and there is much research going on. Yet many problems remain unresolved.

Future work can focus on the provisioning of resource assurance and service differentiation in the context of the target application of this project.

Bibliography

Note: There is an electronic version of the bibliography at
http://www.lancs.ac.uk/postgrad/elkhatib/prj/biblio.htm

[1] IEEE Standard Specification for 802.11, "*Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*", 1999 Edition, Reaffirmed 12 June 2003.

[2] A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, edited by K. Fall, and K. Varadhan, "*The ns Manual (formerly ns Notes and Documentation)*", The VINT Project, July 4 2006
Available at: http://www.isi.edu/nsnam/ns/ns-documentation.html

[3] D. Akin, and J. Geier, "*CWAP (Certified Wireless Analysis Professional) Official Study Guide*", first edition, ISBN: 0072255854, McGraw-Hill/Osborne Media, October 15 2004.

[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "*Wireless sensor networks: a survey*", Computer Networks, 38(4):393-422, March 2002.

[5] B. Bappu, J. Tay, and J. See, "*Location Based Ad-Hoc Message Propagation using WiFi and GPRS Networks*", British Telecommunications plc, 2005.

[6] R. Barr, "*An Efficient, Unifying Approach to Simulation Using Virtual Machines*", a PhD dissertation, Cornell University, May 2004.
Available at: http://jist.ece.cornell.edu/docs/040517-thesis.pdf

[7] P. Barron, S. Weber, S. Clake, and V. Cahill, "*Experiences Deploying an Ad-hoc Network in an Urban Environment*", Trinity College Dublin, 2005.

[8] D. Cavin, Y. Sasson, and A. Schiper, "*On the Accuracy of MANET Simulators*", Proceedings of the ACM Workshop on Principles of Mobile Computing, POMC'02, pages: 38-43, October 2002.

[9] C. Chiang, H. Wu, W. Liu, and M. Gerla, "*Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel*", IEEE Singapore International Conference on Networks, SICON'97, pages: 197-211, April 1997.

[10] J. Chen, S.-H. Gary Chan, J. He, S.-C. Liew, "*Mixed-Mode WLAN: The Integration of Ad Hoc Mode with Wireless LAN Infrastructure*", IEEE Globecom 2003, December 2003.

[11] S. Cherry, "*WiMax and Wi-Fi, separate and unequal*", IEEE Spectrum Magazine, March 2004, page16.

[12] I. Chlamtac, M. Conti, and J. J.-N. Liu, "*Mobile ad hoc networking: imperatives and challenges*", Ad Hoc Networks Journal, Elsevier, vol. 1, no. 1, pages 13--64, July 2003.

[13] S.K. Das, B.S. Manoj, and C.S.R. Murthy, "*A Dynamic Core Based Multicast Routing Protocol for Ad hoc Wireless Networks*", Proceedings of MobiHoc, 2002.

[14] A. Dimakis, L. He, J. Musacchio, H. S. W. So, T. Tung and J. Walrand, "*Adaptive Quality of Service for a Mobile Ad Hoc Network*", The Fifth IFIP TC6 International Conference on Mobile & Wireless Communication Networks, Singapore, October 2003.

[15] R. Dube, C.D. Rais, K. Wang, and S.K. Tripathi, "*Signal Stability Based Adaptive Routing (SSR alt SSA) for Ad Hoc Mobile Networks*", IEEE Personal Communication, February 1997.

[16] M. Ergen, "*IEEE 802.11 Tutorial*", University of California Berkeley, June 2002.
Available at: http://www.eecs.berkeley.edu/~ergen/docs/ieee.pdf
and: http://citeseer.ist.psu.edu/536785.html

[17] E. Ferro, and F. Potortì, "*Bluetooth and Wi-Fi Wireless Protocols: A Survey and a Comparison*", IEEE Wireless Communications, Volume: 12 Issue: 1, Pages: 12-26, 2005.

[18] D. Grigoras, "*Service-oriented Naming Scheme for Wireless Ad Hoc Networks*", University College Cork, 2005.

[19] D. Johnson, D. Maltz, and Y. Hu, "*The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks*", Internet Draft, July 2004.
Available at:
http://www.cs.cmu.edu/~dmaltz/internet-drafts/draft-ietf-manet-dsr-09.txt

[20] R. Khalili, and K. Salamatian, "*Evaluation of Packet Error Rate in Wireless Networks*", Proceedings of the ACM MSWiM 2004, Venice, October 4-6, 2004.

[21] N. Kim, S. Kang, Y. Lee, and B. Lee, "*Name-Based Autoconfiguration for Mobile Ad hoc Networks*", ETRI Journal, vol.28, no.2, pages: 243-246, April 2006.

[22] C. Mallanda, A. Suri, V. Kunchakarra, S.S. Iyengar, R. Kannan, and A. Durresi, "*Simulating Wireless Sensor Networks with OMNeT++*", Department of Computer Science, Louisiana State University, 2005.
Available at:
http://bit.csc.lsu.edu/sensor_web/final_papers/SensorSimulator-IEEE-Computers.pdf

[23] A.J. McAuley, and K. Manousakis, "*Self-Configuring Networks*," 21st Century Military Communications Conference Proceedings, vol.1, pages: 315-319, October 2000.

[24] S. Murthy, and J.J. Garcia-Luna Aceves, "*An Efficient Routing Protocol for Wireless Networks*", ACM/Baltzer Journal on Mobile Networks and Applications (Special Issue on Routing in Mobile Communication Networks) vol. 1, No. 2, pages: 183-197, October 1996.

[25] V. Park, and S. Corson, "*A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks*", IEEE Conference on Computer Communications INFOCOM'97, Volume 3, pages: 1405-1413, April 1997.

[26] T. Parker, and K. Langendoen, "*Guesswork: Robust Routing in an Uncertain World*", 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2005), November 2005.

[27] C. Perkins, E. Belding-Royer, and S. Das, "*Ad hoc On-Demand Distance Vector (AODV) Routing*", RFC 3561, July 2003.

[28] C. Perkins, and P. Bhagwat, "*Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*", ACM SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications, pages: 234-244, 1994.

[29] C. Perkins, E. Royer, and S. Das, "*IP Address Autoconfiguration for Ad Hoc Networks*", Internet draft 2000.
Available at: http://citeseer.ist.psu.edu/298883.html

[30] F. Ros, and P. Ruiz, "*Implementing a New Manet Unicast Routing Protocol in NS2*", Department of Information and Communications Engineering, University of Murcia, December 2004.
Available at: http://masimum.dif.um.es/?Documents

[31] E. Royer, and C.-K. Toh, "*A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks*", IEEE Personal Communications, April 1999.

[32] R. Sivakumar, P. Sinha, and V. Bharghavan, "*CEDAR : A Core-Extraction Distributed Ad hoc Routing Algorithm*", In Haas et al., Wireless Ad Hoc Networks (S.I.), IEEE Journal Selected Areas Communication 17(8), pages: 1454-1465, 1999.

[33] Y. Sun, and E. Belding-Royer, "*Dynamic Address Configuration in Mobile Ad hoc Networks*", UCSB Technical Report 2003-11, June 2003.

[34] C. Toh, "*A Novel Distributed Routing Protocol To Support Ad hoc Mobile Computing*", Proceedings of IEEE 15th Annual International Phoenix Conference on Computers and Communications, IEEE IPCCC 1996, pages: 480-486, March 1996.

[35] S. Toner, and D. O'Mahony, "*Self-Organising Node Address Management in Ad-hoc Networks*", Trinity College Dublin, 2003.

[36] N. Vaidya, "*Weak Duplicate Address Detection in Mobile Ad hoc Networks*", Proceedings of MobiHoc 2002.

[37] S. J. Vaughan-Nichols, "*Achieving wireless broadband with WiMax*", pages: 129-136, IEEE Computer, June 2004.

[38] J. Yoon, M. Liu and B. Noble, "*Sound Mobility Models*", Proceedings of ACM MobiCom, San Diego, September 2003.

[39] H. Zhou, L Ni, and M. Mutka, "*Prophet Address Allocation for Large Scale MANETs*", Proceedings of the 22nd Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM 2003), San Francisco, April 2003.

[40] Internet Engineering Task Force, "*Mobile Ad-hoc Networks (manet) Charter*" http://www.ietf.org/html.charters/manet-charter.html

[41] WiFi Alliance, http://www.wi-fi.org/

[42] The VINT Project, "*The Network Simulator - ns-2*", University of South California, http://www.isi.edu/nsnam/ns/

[43] Cygwin, "*Cygwin Information and Installation*", http://www.cygwin.com/

[44] SourceForge, "*OTcl and TclCL home*", http://otcl-tclcl.sourceforge.net/otcl/

[45] Wikipedia, "*Ad hoc Routing Protocol List*", version: May 1st 2006, 23:27, by (IP 68.35.252.176), http://en.wikipedia.org/wiki/Ad_Hoc_Routing_Protocol

[46] Wikipedia, "*AWK Programming Language*", version: June 7th 2006, 15:09, by Lost.goblin, http://en.wikipedia.org/wiki/Awk

[47] Autonomic Network Architecture Research Project, http://www.ana-project.org/

[48] BIONETS, "*BIONETS: Bio-inspired Service Evolution for the Pervasive Age*", http://www.bionets.org/

[49] Car2Car Communication Consortium, http://www.car-2-car.org/

[50] DAIDALOS Project, "*FP6 IST Integrated Project DAIDALOS*" http://www.ist-daidalos.org/

[51] DTNRG, "Delay-Tolerant Networking Research Group", http://www.dtnrg.org/

[52] E-NEXT, "*E-NEXT: Network of Excellence Emerging Network Technologies*", http://www.ist-e-next.net/

[53] FleetNet, "*FleetNet - Vision*", http://www.et2.tu-harburg.de/fleetnet/

[54] FOKUS, "FOKUS - der FuE-Partner für Innovationen und offene Kommunikationssysteme", http://www.fokus.fraunhofer.de/home/

[55] Haggle Project, "*Haggle: A European Union funded project in Situated and Autonomic Communications*", http://www.cambridge.intel-research.net/haggle/

[56] IP CASCADAS, "CASCADAS Project", http://www.cascadas-project.org

[57] MobiLife, "*Mobilife: the development web*", http://www.ist-mobilife.org/

[58] NOW, "NOW: Network On Wheels", http://www.network-on-wheels.de/

[59] IBBT SPAMM, "*Project SPAMM: Solutions Platform for Advanced Mobile Mesh*", http://projects.ibbt.be/spamm/

[60] GloMoSim, "*Global Mobile Information Systems Simulation Library*", UCLA Parallel Computing Laboratory, http://pcl.cs.ucla.edu/projects/glomosim/

[61] JiST/SWANS, "*Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator*", School of Electrical and Computer Engineering, Cornell University, http://jist.ece.cornell.edu/

[62] NCTUns, "NCTUns 3.0 Network Simulator and Emulator", Network and System Laboratory, Department of Computer Science and Information Engineering, National Chiao Tung University Hsinchu, Taiwan, http://nsl.csie.nctu.edu.tw/nctuns.html

[63] OMNeT++, "*OMNeT++ Community Site*", http://www.omnetpp.org/

[64] OPNET, "*Making Networks and Applications Perform*", http://www.opnet.com/

[65] Prowler, "ISIS – Prowler - Probabilistic Wireless Network Simulator", NEST (Network Embedded Systems Technology) Project, Institute for Software Integrated Systems, Vanderbilt University, Nashville, Tennessee, http://www.isis.vanderbilt.edu/Projects/nest/prowler/

[66] QualNet, "*SNT: QualNet Family of Products*", Scalable Network Technologies (SNT), http://www.scalable-networks.com/products/qualnet.php

[67] BBC, "*City First For Wi-Fi Connection*"
http://news.bbc.co.uk/1/hi/england/lancashire/3541346.stm
Published August 6 2004, accessed August 18 2006

[68] BBC, "*City-Wide Wi-Fi Rolls Out In UK*"
http://news.bbc.co.uk/1/hi/technology/4578114.stm
Published January 3 2006, accessed May 6 2006

[69] C-Net, "*Microsoft's Zune Aims to Be Social Butterfly*"
http://news.com.com/2100-1041_3-6109667.html?part=rss&tag=6109667
Written by Ina Fried, published August 25 2006, accessed August 28 2006

[70] eWeek, "*T-Mobile Opens Wireless Floodgates in New Orleans*"
http://www.eweek.com/article2/0,1759,1854453,00.asp?kc=EWRSS03119TX1K0000594
Written by Larry Loeb, published August 31 2005, accessed August 18 2006

[71] GeekZone, "*Sony Introduces mylo Pocket Communicator*"
http://www.geekzone.co.nz/content.asp?contentid=6531
Published August 8 2006, accessed August 28 2006

[72] Light Reading, "*Philly City Council OKs WiFi Plan*"
http://www.lightreading.com/document.asp?doc_id=94758
Written by Carmen Nobel, published May 12 2006, acessed May 13 2006

[73] Orlando Sentinel, "*Downtown Orlando Will Surf for Free*"
http://www.orlandosentinel.com/orl-wifi2906apr29,0,2079383.story?coll=orl-home-headlines
Published April 29 2006, accessed May 6 2006

[74] SF Gate, "*Google Offers S.F. Wi-Fi*"
http://www.sfgate.com/cgi-bin/article.cgi?file=/c/a/2005/10/01/MNGG9F16KG1.DTL
Written by Verne Kopytoff and Ryan Kim, published October 1 2005, acessed May 6 2006

[75] Spotlighting News, "*New York City Wants to Have Wireless Internet by July in Central Park*"
http://www.spotlightingnews.com/article.php?news=2201
Published May 16 2006, accessed May 16 2006

[76] Taipei City Government, "*Public Wireless Local Network (PWLN) "*
http://english.taipei.gov.tw/TCG/index.jsp?recordid=1994
Published November 16 2004, accessed February 11 2006

[77] TMCnet, "*Citywide Wifi Could Bring Cyberspace at a Lower Cost*"
http://www.tmcnet.com/usubmit/2006/04/17/1577653.htm
Written by Alexis Grant, published April 17 2006, accessed April 23 2006

[78] TMCnet, "*Net2Phone Taps BridgePort to Offer MobileVoIP*"
http://news.tmcnet.com/news/-net2phone-bridgeport-mobile-voip-dual-mode-cellular-/2006/05/09/1642050.htm
Written by Johanne Torres, published May 9 2006, accessed May 13 2006

[79] United Press International, "*MetroFi selected for WiFi in Portland*"
http://www.upi.com/Hi-Tech/view.php?StoryID=20060414-022910-1447r
Published April 14 2006, acessed May 13 2006

Appendix A:

The Proposal for the Project

The following pages contain the proposal submitted on the 19th of May, 2006. Please note that the proposal has a separate list of references (included therewith) not to be confused with the bibliography of this report.

WiFi Ad Hoc Message Propagation Over GPRS Networks

Abstract

Ad hoc wireless networks extend the reach of networks beyond the geographical location of any infrastructure (defined by access points). Such networks can be very useful in commercial, military, intelligence collection, and personal applications. In this report, a proposed project is presented that emphasises on propagating messages, in form of commercial adverts, across an arbitrary ad hoc wireless network. The first main aim of the project is to develop an addressing scheme that is versatile in avoiding possible conflicts in such unsystematic networks. The other aim is to establish a routing protocol that can rapidly provide paths across the network that help optimise the throughput and delay. The project should provide a commercial solution that is also applicable to other fields.

1 Introduction

Wireless networks have become quite popular over the last few years for the convenience, reliability, and bandwidth. Wireless adaptors and access points are now widely availble at low prices. More importantly, there are no interoperability problems inbetween products of different vendors since they are all standardised by the WiFi Alliance [18].

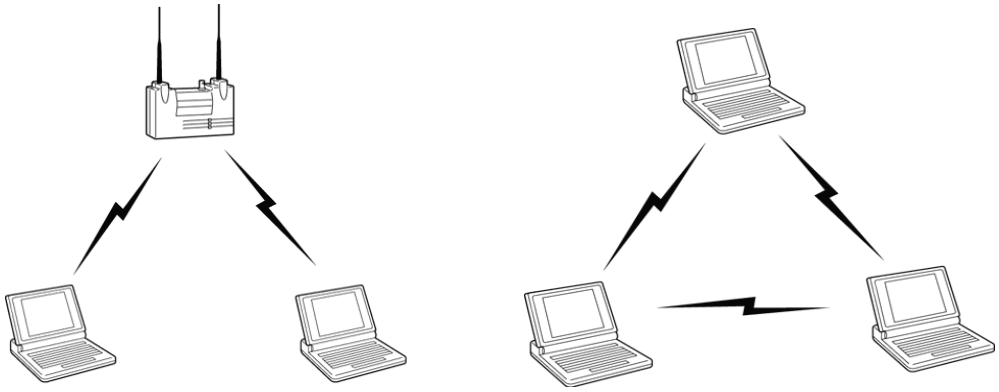
WiFi is now broadly adopted in houses, public places, and corporations. Users prefer WiFi-enabled devices, whether be it desktop computers, laptops, PDAs, or mobile phones. At the same time wireless access points are being installed in a race to attract potential users. Tens of WiFi hot zones are to be set up in all major UK cities [7] to provide city-wide *WiFi blankets*. The same is happening in many cities outside the UK as well, such as in Taipei [13], Houston [14], San Francisco [11], Philadelphia [9], Portland [16], Orlando [10], and New York City's Central Park [12].

The mobile industry is a huge one with increasing potential. The number of unique mobile users in the UK exceeded 43 million, which is over 82% of the population [17]. Within this enormous number of users, WiFi is gradually gaining the same popularity it has in the computer world. The availabaility of access points has encouraged users to invest in WiFi-enabled mobile phones, which there is no shortage of. Currently there are 29 different WiFi-enabled mobile phones available on the market (listed in the appendix). Several attempts are being done by telecom providers to encourage their users to use WiFi-enabled phones, such as the project between Net2Phone and BridgePort Networks to provide VoIP to mobile phones through WiFi [15].

However, when it comes to WiFi most users only use it for short-period Internet access when they are away from home and out of their offices. The majority of the time, the WiFi adaptor on their devices is inactive and most of the bandwidth goes unutilised. Rooting from this concern and with a potential marketing opportunity, British Telecom has proposed the utilisation of WiFi devices to propagate messages to other devices in the same area. All economical reasons aside, the main objective of

this project is to exploit the unused WiFi bandwidth by using available wireless devices in ad hoc mode.

Wireless devices can be connected to each other in one of two ways: *infrastructure* and *ad hoc*. In infrastructure mode, all wireless devices are connected to an access point and hence all communication goes through that. On the other hand, devices in ad hoc mode are connected to each other without any need for an access point. The nodes in ad hoc mode cooperate to organise and maintain the network inbetween them. Figure 1 below illustrates the two WiFi modes.



Infrastructure mode Ad hoc mode

Figure 1. The two modes of WiFi
(images from http://support.dell.com/support/edocs/network/079nk/specs.htm)

Infrastructure mode is the more common of the WiFi modes. As a matter of fact, most WiFi users are not at all familiar with ad hoc mode that they, for example, rely on floppy or USB disks to exchange files.

The rest of this report is ordered as follows: section 2 sets the scene of ad hoc wireless networks, and gives an overview of the goal scenario. It then gives a closer look on the scenario and its complications. This section also mentions a couple of projects related to the proposed one. Section 3 introduces the proposed project by listing the aims of the project, and by describing the methodology used to achieve those aims. Section 4 sets the tasks to be completed and the schedule which will be followed in the completion of the tasks. Section 5 mentions the resources required for the project.

2 Background

There are a few concerns when it comes to the field of ad hoc wireless networks. First of all, there is no standard naming system. Hence, there is no way to translate a name into an address. Furthermore, no IP address configuration services, such as DHCP, are offered. Since all nodes in ad hoc mode are equal, there is no such server or gateway to keep and maintain IP address configurations.

Another major concern about ad hoc wireless networks is that they are typically quite dynamic. Nodes in such networks are arbitrarily located and tend to keep moving in unpredictable directions most of the time. Moreover, ad hoc wireless networks can extremely grow or shrink in size in a very short period of time. Therefore, the routing protocol to be used in such networks should be robust and scalable. Such a protocol is expected to effictively maintain the routing table, and to

find routes that not just minimise the delay and maximise the throughput, but also minimise the power consumption.

The way that these issues are handled will depend on the specifics of the envisaged scenario. The following Figure depicts an example scenario provided by BT, and one which I have chosen as the target scenario. There are a few other scenarios suggested by BT, but this one in particular illustrates the problem in a simple way. All the other scenarios are slight variations of this one.

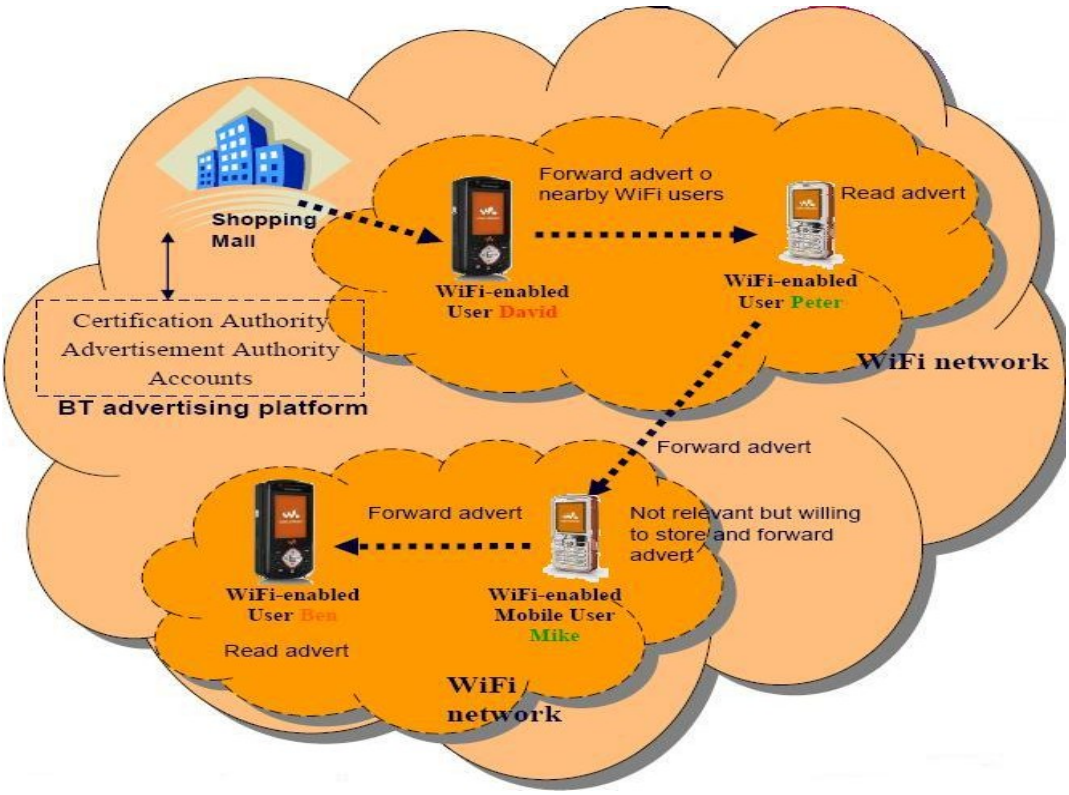


Figure 2. High Street Adverts Scenario

2.1 High Street Adverts Scenario

A simple description o f the scenario is as follows: User David, for example, is walking down the street. As he passes by a BT WiFi hotzone, the access point recognises his device as a customer device and sends him one or more adverts sponsored by some of the shops on the street. As he passes other users with WiFi-enabled devices, the adverts are forwarded to them, and so on. Users will be motivated to forward the messages by means of incentives offered either by the provider (BT) such as free minutes, or by the advertiser such as discounts.

2.2 Analysis

In the simple scenario mentioned above, there are a few unanswered questions. The first device to recieve a particular advert from an acess point will of course be in infrastructure mode. Then it goes into ad hoc mode to forward the advert to other devices, but is the switch between infrastructure and ad hoc mode simple? Is it automatically done, or does it need interaction from the user? How long does it take?

Notice that the communication between the devices is anonymous, in the sense that there is no need for the sender or the receiver to know each other. The sender

simply offers the message, and the receiver chooses to accept it or not without any knowledge of the sender. The receiver only cares whether the offered message is genuine or not. Thus, no naming service is required. The message (advert in this case) is forwarded to any reachable device whose user is willing to accept it.

Yet there is still a need to know the devices' addresses. Each device must be claim an IP address without causing conflicts with other devices around, taking into consideration the constantly changing nature of ad hoc wireless network topologies. One possible option is randomisation, which might prove to be a tricky task. Another option is DHCP via GPRS. The devices can consult to a server using GPRS connections. The problem with this scheme is that establishing a GPRS connection takes a relatively long time. A few other suggestions have been made in this field. In one of them, Toner and O'Mahony suggest an autoconfiguration addressing scheme for ad hoc wireless networks [5].

It must also be noted that no addressing scheme might be needed at all if we can propagate the messages using broadcast on the link layer. The issue of addressing is of great importance, and can only be settled after deliberate research into the 802.11 standards, and studying of the target message exchange protocol.

A further major obstacle is routing. A routing protocol is required to discover nearby devices, update the routing table as devices join and leave the vicinity, and find the best routes to optimise throughput, delay, and power consumption. This protocol must be robust in the face of rapid topology changes. Several routing protocols have been designed for mobile ad hoc networks, but each has its strengths and flaws.

The last, but not least, of the problems to tackle is the control over message propagation. There are many concerns in the propagation of messages. The users will have profiles to describe the type of adverts they might be willing to accept. Messages contain metadata to classify according to the details. The metadata will also contain the lifetime of the message among other things. The main concern here is to avoid duplicate messages. No user should recieve any particular advert more than once. Additionally, the mechanism should react gracefully to connections which fail due to topology changes.

There are difinitely a lot of other issues that might come up as the work advances, but these are the major ones at the current moment.

2.3 Similar Work

ANA is a joint project between Lancaster University and several other academic and commercial European institutions [6]. It is a project that aims at expanding the Internet beyond its traditional technologies by incorporating resources in the form of mobile devices in ad hoc wireless networks. The goal is a network that adapts itself according to demands upon it.

Haggle is a project by Intel and Cambridge to manage and share resources across ad hoc mobile networks formed by personal WiFi-enabled devices [8]. As it is described on the project's homepage, the project corresponds to a Google of the ad hoc networks. Needles to say, there is a strong connection between these two projects and the one proposed by this report. Message propagation forms the basis of both ANA and Haggle.

3 Proposed Project

3.1 Aims

The objective of the project is to exploit unused wireless bandwidth by using available wireless devices in ad hoc mode to propagate messages to other devices in the same area. The following has been identified as the main aims of the work:

1. Define a scheme to assign IP addresses to the wireless devices while avoiding conflicts, and to resolve any conflict if they arise.
2. Define a routing protocol that can perform efficiently in an ad hoc wireless environment.
3. Develop a message exchange mechanism to fit in with the application requirements.

3.2 Methodology

Substantial research into the 802.11 and GPRS standards is necessary before any decision is to be made about how to safely assign IP addresses. Initial literature search will be about the possibility of avoiding IP addresses by using link layer broadcast. If, however, it is found out that logical addressing is a necessity, the investigation will be focused upon the possible addressing techniques aforementioned (i.e. randomisation, DHCP via GPRS, etc.).

The routing issue should also be handled with care after examining the available protocols. Many of these protocols were optimised for a particular case (e.g. unidirectional links). If a particular protocol is found to fit our scenario, then it may be used as it is or after slight modification. The protocol should be evaulated on basis of the chosen routes in terms of throughput, delay, and power consumption. Optimising thoughput and delay will be the primary goal, while optimising power consumption will come as a secondary goal which will only be attempted to achieve if time allows.

The messaging mechanism will basically have to exchange the list of available messages with neighbouring nodes. Receiving users should be prompted to accept or deny the messages. Upon approval, the whole content of messages should be transferred. The mechanism should only allow genuine messages to be transferred, while denying any fraud messages (spam). All messages will be digitally signed, and hence can be checked for integrity using GPRS. The implementation of this mechanism, however, is not one of the aims of this project. Instead, an assumption will be made that a PKI (or a similar infrastructure) is put in place. A certificate authority or any other trusted authority will be assumed to be established, and will be referred to (using GPRS) in order to authenticate and check the integrity of all messages. The mechanism should also avoid any message to be received by the same user more than once. It also desirable to allow information about the message propagation to be sent to an assumed tracking server. This information will help the provider (hosting the server) to retrieve accounting information. It could also be quite valuable to evaluate the message propagation mechanism. Such feedback could help in optimising the initial dispersion of the messages (amount of initial messages, graphical location of initial disperse, time of the day, day of the week, user groups, etc.).

3.3 Testing & Evaluation

The application will be deployed and evaluated using a simulator. There is a number of available network simulators, such as ns-2, JiST/SWANS, Prowler, OMNet, and SimReal. The choice of which simulator to use will be made once the previous issues have been properly addressed since the chosen simulator must support the decisions taken regarding the previous issues. Analysis of the system behaviour

4 Programme of Work

4.1 Tasks

The tasks that should be accomplished in order to implement this project are listed below. Each task is briefly described, and a rough estimate is made about the time required to finish it. These tasks are based on my current understanding of the field of ad hoc mobile networks, and may change in compliance with the research.

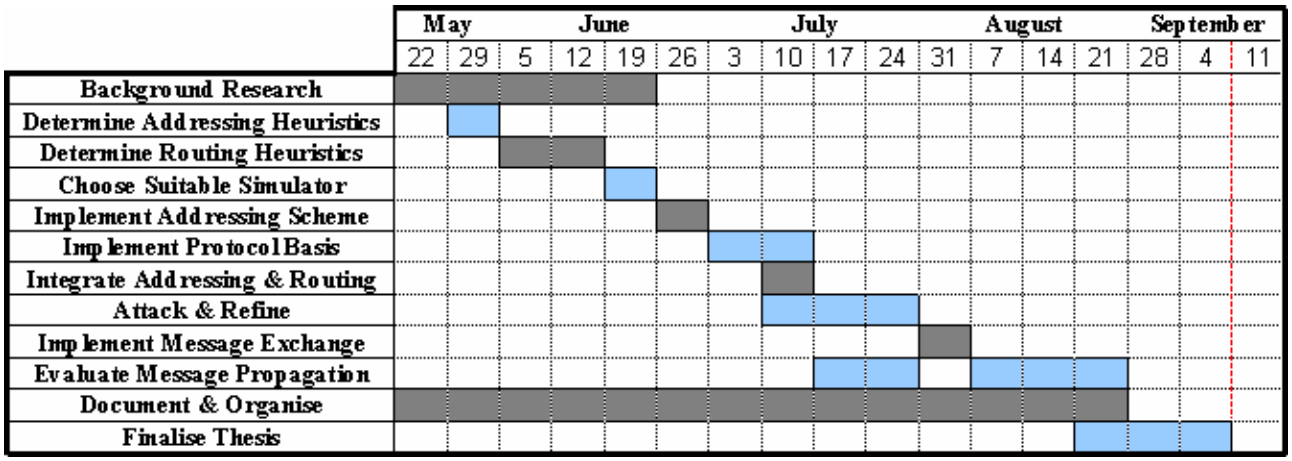
1. Study about the 802.11 and GPRS standards: The aim here is to have a sufficient understanding of the protocols and restrictions of using wireless networks and GPRS. This task has already started, and understanding of the basics would probably be achieved by the first week of June. Needless to say, this research will continue through the summer, but this task refers to the initial, intense literature reading phase.
2. Determine the heuristics of addressing: This task will be achieved through research focused on the problems of addressing in ad hoc wireless networks and the requirements of the target application. This has also already started, but will still need at least one more week to reach a state where a sensible decision can be made.
3. Determine the heuristics of routing: Similar to the previous task, this will focus on routing in ad hoc mobile networks. This shall take between one and two weeks.
4. Examine the available simulators and choose one which suits the requirements: The available netowrk simulators will be studied and examined. In the light of the heuristics of addressing and routing, a suitable one will be picked. This shall not take more than a few days, and a couple more to familiarise myself with the chosen simulator package.
5. Implement an addressing scheme: The decided addressing scheme will be implemented using the chosen simulator. I have set one week for this task.
6. Implement the basis of a routing protocol: similarly, the routing protocol will be implemented.
7. Integrate the implementations of the addressing scheme and the routing protocol: One week to review the implementations and integrate them together.
8. Attack the implementation under different circumstances, and refine accordingly: The implementation will be tried out in simulated scenarios, and tested under different circumstances. Changes to the implementation are expected here. Hence, I have set a loose period of time of three weeks for this task. This also provides a few extra days if the previous 3 tasks exceed their schedule. Alternatively, if all goes well then these few extra days may be used as a short break.
9. Implement the message exchange mechanism: Implementation of the higher

level mechanism to control the propagation of messages through the network. This is a relatively easy task, and should not take more than a week to accomplish.

10. Evaluate the whole message propagation application: This is a vital task where the implementation as a whole is put to the test to see what can be achieved by the application. As mentioned, the application will be tested on a simulated environment. Main outcomes are throughput and delay analysis in as many different circumstances as possible. Analysis of the initial message dispersion, power consumption, and range are secondary outcomes.
11. Documentation: This task should be carried out throughout the whole project. During the backround research stage a summary will be written about the studied literature on a daily basis. The design and evaluation of the addressing scheme and routing protocol will also be noted on as progress is made. Finalisation of the document will start as the evaluation phase reaches its end, and will be over by the first days of September 2006.

4.2 Schedule

The project is scheduled to be complete in 15 weeks. The following chart shows a rough schedule for the completion of each of the tasks over this period.



5 Resources

The resources that shall be required to complete the project include:

- network simulator: Most simulators, such as ns-2 and OMNet, use a well-known object-oriented programming language (typically C++) for developing modules and another high-level programming language as a frontend. Obviously, a C++ compiler is also required.
- operating system: Most simulators run on both Unix and Windows.

6 References

[1] B. Bappu, J. Tay, and J. See, "*Location Based Ad-Hoc Message Propagation using WiFi and GPRS Networks*", British Telecommunications plc., 2005

[2] S.K. Das, B.S. Manoj, and C.S.R. Murthy, "*A Dynamic Core Based Multicast Routing Protocol for Ad hoc Wireless Networks*", Proceedings of MobiHoc, 2002

[3] D. Grigoras, "*Service-oriented Naming Scheme for Wireless Ad Hoc Networks*", University College Cork, 2005

[4] E.M. Royer, and C.-K. Toh, "*A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks*", IEEE Personal Communications, April 1999

[5] S. Toner, and D. O'Mahony, "*Self-Organising Node Address Management in Ad-hoc Networks*", Trinity College Dublin, 2003

[6] ANA: Autonomic Network Architecture
http://www.personnel.lancs.ac.uk/furtherinformation.aspx?detailsid=A631.html
accessed May 13th, 2006

[7] BBC, *City-wide wi-fi rolls out in UK*, January 3rd, 2006
http://news.bbc.co.uk/1/hi/technology/4578114.stm
accessed May 6th, 2006

[8] Haggle Project
http://www.cambridge.intel-research.net/haggle/index.php
accessed May 13th, 2006

[9] Light Reading, *Philly City Council OKs WiFi Plan*, May 12th, 2006
http://www.lightreading.com/document.asp?doc_id=94758
acessed May 13th, 2006

[10 Orlando Sentinel, *Downtown Orlando will surf for free*, April 29th, 2006
http://www.orlandosentinel.com/orl-wifi2906apr29,0,2079383.story?coll=orl-home-headlines
accessed May 6th, 2006

[11] SF Gate, *Google offers S.F. Wi-Fi. ..*, October 1st, 2005
http://www.sfgate.com/cgi-bin/article.cgi?file=/c/a/2005/10/01/MNGG9F16KG1.DTL
acessed May 6th, 2006

[12] Spotlighting News, *New York City wants to have Wireless Internet by July in Central Park*, May 16th, 2006
http://www.spotlightingnews.com/article.php?news=2201
accessed May 16th, 2006

[13] Taipei City Government, *Public Wireless Local Network (PWLN)*, April 17th, 2006
http://english.taipei.gov.tw/TCG/index.jsp?recordid=1994
accessed February 11th, 2006

[14] TMCnet, *Citywide WiFi could bring cyberspace at a lower cost*, April 17th, 2006
http://www.tmcnet.com/usubmit/2006/04/17/1577653.htm
accessed April 23rd, 2006

[15] TMCnet, *Net2Phone Taps BridgePort to Offer MobileVoIP*, May 9th, 2006
http://news.tmcnet.com/news/-net2phone-bridgeport-mobile-voip-dual-mode-cellular-/2006/05/09/1642050.htm
accessed May 13th, 2006

[16] United Press International, *MetroFi selected for WiFi in Portland*, April 14th, 2006
http://www.upi.com/Hi-Tech/view.php?StoryID=20060414-022910-1447r
acessed May 13th, 2006

[17] W2Forum: "*UK mobile ARPU will continue to climb*"
http://www.w2forum.com/i/UK_Telecoms_Statistics/
accessed May 11th, 2006

[18] WiFi Alliance
http://www.wi-fi.org/

Appendix B:

IEEE 802.11 Specifications Overview

This section gives an overview of the IEEE 802.11 specifications, focusing mainly on the MAC (Media Access Control) layer.

B.1 MAC Traffic Control

The 802.11 MAC layer implements mechanisms for physical carrier sense, collision detection, and collision avoidance. Nodes only commence transmission when they can detect that no other node in their range is transmitting. However, for nodes that are out of the transmitting node's range it is not possible to detect whether they are currently receiving packets or not. This is known as *the hidden node problem*. To address this problem, 802.11 employs an additional mechanism to govern the transmission of unicast traffic. RTS/CTS (Request to Send/Clear to Send) is a straightforward handshake mechanism where the transmitter sends a RTS message and waits for the receiver to reply with a corresponding CTS message. The receiver will only respond when it is idle. Once the CTS message is received, the transmitter sends a packet.

RTS/CTS also provides a solution to *the exposed node problem*. Consider an example where nodes A and B lie within each other's range, as in Figure B.1. Node A wants to transmit to node AA while node B is currently transmitting to node BB. AA is not within B's range, and BB is not within A's range. Although traffic between A and AA would not cause interference in the connection between B and BB, A is prevented from transmitting as it detects B's traffic. RTS/CTS solves this issue: If A can detect B's RTS messages but no corresponding CTS replies, then it assumes that it is safe to commence transmission to AA.

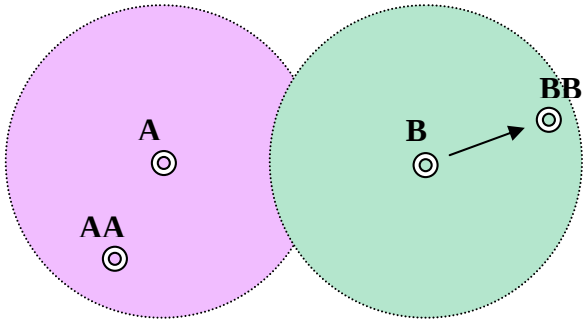


Figure B.1: The Exposed Node Problem

The RTS/CTS procedure is repeated for every data packet. However, to reduce the overhead caused by the RTS and CTS messages, 802.11 defines a parameter RTS

threshold. This parameter defines the maximum size of a packet which can be sent without using RTS/CTS.

The 802.11 MAC layer uses simple carrier sense for broadcast traffic, and both carrier sense and RTS/CTS for unicast traffic. However, those techniques are not enough to ensure collision-free WiFi communication. Collisions could still occur as a result of simultaneous transmissions, or due to interference in the wireless channel. For these problems, the 802.11 MAC requires acknowledgement for each transmitted packet. If no acknowledgement is received by the transmitter for a particular timeout period, the transmitter backs off (exponentially) and then resends the packet. The transmitter keeps trying to resend the packet as long as no acknowledgement is received. The maximum number of trials is a modifiable parameter called the short/long retry limit, depending on the packet size. This is known as the 802.11 ACK mechanism.

B.2 Frame Format

We shall only show the 802.11 frame format, but without mentioning any details since it was seen irrelevant to the project.

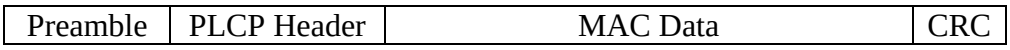


Figure B.2: The 802.11 Frame Format

The PLCP (Physical Layer Convergence Procedure) header contains information that is used by the physical layer to decode the frame, such as the length of the MAC data and the signalling rate.

B.3 Fragmentation & Reassembly

Since wireless links are more prone to interference and noise than conventional wired links, it is preferable to use smaller packets in transmission. In case retransmission is necessary, the added overhead is reduced for smaller packets. Moreover, there is a higher probability of a packet getting altered during the transmission of larger packets. Therefore, the 802.11 MAC layer performs additional fragmentation/reassembly. Packets are broken down into smaller fragments, which are individually acknowledged by the receiver.

B.4 Synchronisation & Beaconing

Nodes of a wireless local area network (LAN) keep sending periodic frames, called beacon frames. These serve two main functions. First, they facilitate more reliable communication by providing a mechanism to keep the clocks of the nodes synchronised. In infrastructure mode, only the wireless access point beacons while the served nodes listen and update their clocks. In ad hoc mode, the node that starts the network would start beaconing and will be responsible for setting the beaconing period. Other nodes that join the network will set their beacon timeout period accordingly, and will attempt to send beacon frames when the timeout expires. If they detect a beacon in this period they will cancel their beacon and will back off for a random period of time. In both modes, beacon frames that are lost due to collision are not retransmitted.

The other importance of the beaconing process is that it helps in identifying the nodes that lie in range by recognizing the presence of new nodes, and the absence of known nodes.

Beacon frames serve additional functions. Low-power operation is an operation mode that can be used to conserve the battery power of nodes. Using this mode, nodes can sleep unless they need to transmit or receive frames. This is particularly useful for nodes of relatively expensive battery power, such as nodes in sensor networks [4]. In such situations, the beacon frames could be used to convey information about the data frames that need to be buffered for transmission. Beacon frames are also used to send across the capability of nodes, such as physical layer parameters and supported rates.

Appendix C:

Operating Both WiFi Modes Simultaneously

802.11 does not support autonomic switching between ad hoc and infrastructure modes. Here, we discuss two projects that are concerned with finding a means, which is autonomic, graceful, and transparent to the user, of switching between the 802.11 modes.

Mixed-Mode is a framework where both modes are in use by different nodes at the same time [10]. When too many nodes join a wireless cell managed by an access point, the contention and collision levels in the wireless channel can become too high. Using Mixed-Mode, the access point can instruct a group of served nodes to switch to ad hoc mode. However, the switch has to be swift so as to avoid any interruption in any established connections. The switch also needs to be transparent to the user. The other nodes still operating in infrastructure mode will share the connection to the access point with the ad hoc nodes. The reduction in the number of nodes served by an access point prevents throughput degradation significantly, which is the main aim of the Mixed-Mode project.

The SPAMM Project [59] is also concerned with operating in the two modes of WiFi. The project is still ongoing and it aims at providing wireless communication to vehicles equipped with WiFi adaptors using three different alternatives. Initially, a vehicle tries to use any WiFi infrastructure available, such as access points in hot-spots. If no such coverage is available, the vehicle looks for other vehicles within its wireless range. If one is found, the WiFi adaptor would switch to ad hoc mode. If neither WiFi options are available, then the vehicle can use small-band networks such as GPRS and UMTS.

One of the main research areas of the SPAMM project, as explained on the project's website, aims to establish means for switching between infrastructure WiFi, ad hoc WiFi, and GPRS/UMTS. Such a switch should be autonomous and completely transparent to the user.

Appendix D:
Routing Protocols Simulation Results

According to the explanations of DSR and AODV in sections 2.3.4 and 2.3.5, respectively, the routing overhead of DSR is more than that of AODV. This is because DSR produces an accumulative route and considers multiple paths and unidirectional links. In networks where only single paths are sufficient and where all links are bidirectional, the overhead produced by DSR becomes an unnecessary burden. Nevertheless, we have set up a series of simulations to compare between AODV and DSR beyond theory.

D.1 The Simulation Model

We have created two simulated networks of different sizes. For each network, we have tested it in six different node movement scenarios. Two parameters make these scenarios different from each other:

- average pause time
- maximum speed

These scenarios were used to produce a dynamic topology similar to that of the target network. The topology of the target network is expected to change continuously not only because of node mobility, but also due to radio interference, obstacles (such as buildings), etc. For the lack of appropriate means of representing such changes in our simulations, we have amplified the mobility parameters slightly.

The six different node mobility scenarios are as follows:

Scenario	Average Pause Time	Maximum Speed
I	0.5 seconds	1 meter/second
II	0.5 seconds	2 meter/second
III	0.5 seconds	5 meter/second
IV	1.0 seconds	1 meter/second
V	2.0 seconds	1 meter/second
VI	5.0 seconds	1 meter/second

Table D.1: The Mobility Scenarios Used for Comparing DSR with AODV

The first network contained 20 nodes using TCP flows to intercommunicate. 12 flows have been modelled randomly between the nodes. The connection pattern has been saved in the file “tcp-20.12”, available from the website. The second network contained 50 nodes, with 30 TCP flows. The corresponding pattern file is “tcp-50.30”. Both networks run in an area of 500 by 500 meters.

D.2 Results

The graphs in section 2.3.6 were plotted from the following detailed statistics:

D.2.1 Network of 20 Nodes Using DSR

	Scenario	I		II		III	
	pause time	0.5 seconds					
	maximum speed	1 m/s		2 m/s		5 m/s	
		Bytes	*packets*	*Bytes*	*packets*	*Bytes*	*packets*
Routing	average routing packet size	303.44		302.97		302.44	
Routing	sent	6921972	22812	8065000	26620	7586932	25086
Routing	overhead	151.55%	148.06%	107.34%	104.42%	120.72%	117.38%
		seconds		*seconds*		*seconds*	
Delay	maximum	6.38362		5.45555		31.32937	
Delay	minimum	0.00194		0.00194		0.00194	
Delay	average	0.54059		0.44262		0.49126	
Throughput		*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*
Throughput	maximum	20046.41	20046.41	21261.36	21261.36	20005.22	20005.22
Throughput	minimum	1201.60	13.95	503.61	12.59	1043.64	14.39
Throughput	average	8342.42	27.99	9445.00	31.00	9568.40	31.46
Throughput	average (KBps)	8.15		9.22		9.34	

	Scenario	IV		V		VI	
	pause time	1 sec		2 sec		5 sec	
	maximum speed	1 m/s					
		Bytes	*packets*	*Bytes*	*packets*	*Bytes*	*packets*
Routing	average routing packet size	303.81		304.66		305.02	
Routing	sent	7296868	24018	6920156	22714	8006580	26249
Routing	overhead	135.85%	131.77%	151.77%	147.38%	110.19%	105.52%
		seconds		*seconds*		*seconds*	
Delay	maximum	5.03857		3.83308		5.55508	
Delay	minimum	0.00194		0.00194		0.00194	
Delay	average	0.56171		0.60604		0.42908	
Throughput		*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*
Throughput	maximum	17845.31	17845.31	24192.43	24192.43	27417.29	27417.29
Throughput	minimum	990.06	7.00	1014.06	11.22	577.63	7.92
Throughput	average	8638.31	28.82	8246.12	27.97	9366.75	31.38
Throughput	average (KBps)	8.44		8.05		9.15	

D.2.2 Network of 20 Nodes Using AODV

	Scenario	I		II		III	
	pause time	0.5 seconds					
	maximum speed	1 m/s		2 m/s		5 m/s	
		Bytes	*packets*	*Bytes*	*packets*	*Bytes*	*packets*
Routing	average routing packet size	293.56		275.33		266.43	
	sent	4209708	14340	4047352	14700	4128316	15495
	overhead	82.04%	85.58%	2.27%	10.78%	31.37%	46.93%
		seconds		*seconds*		*seconds*	
Delay	maximum	3.41396		3.74287		6.72668	
	minimum	0.00194		0.00194		0.00194	
	average	0.66161		0.38668		0.38601	
Throughput		*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*
	maximum	15536.56	15536.56	22015.19	22015.19	23778.96	23778.96
	minimum	1155.66	14.34	356.19	5.94	1053.54	11.62
	average	7256.73	22.13	10050.86	30.59	9187.10	28.42
	average (KBps)	7.09		9.82		8.97	

	Scenario	IV		V		VI	
	pause time	1 sec		2 sec		5 sec	
	maximum speed	1 m/s					
		Bytes	*packets*	*Bytes*	*packets*	*Bytes*	*packets*
Routing	average routing packet size	280.38		284.49		273.36	
	sent	4098076	14616	4046296	14223	4070896	14892
	overhead	34.83%	43.52%	59.83%	67.70%	5.42%	15.18%
		seconds		*seconds*		*seconds*	
Delay	maximum	4.03543		5.43996		3.56248	
	minimum	0.00194		0.00194		0.00194	
	average	0.56162		0.62743		0.43562	
Throughput		*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*
	maximum	18362.85	18362.85	23552.28	23552.28	28392.79	28392.79
	minimum	1511.41	10.65	1260.89	10.49	1282.70	10.10
	average	8992.95	27.93	7973.22	24.41	10608.45	33.01
	average (KBps)	8.78		7.79		10.36	

D.2.3 Network of 50 Nodes Using DSR

	Scenario	I		II		III	
	pause time	0.5 seconds					
	maximum speed	1 m/s		2 m/s		5 m/s	
		Bytes	*packets*	*Bytes*	*packets*	*Bytes*	*packets*
Routing	average routing packet size	278.67		286.47		276.9	
Routing	sent	13246432	47534	12816316	44738	13622752	49197
Routing	overhead	119.94%	134.68%	126.85%	136.36%	114.83%	131.48%
		seconds		*seconds*		*seconds*	
Delay	maximum	43.81552		14.69672		35.38312	
Delay	minimum	0.00194		0.00194		0.00194	
Delay	average	0.73000		0.79938		0.64475	
Throughput		*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*
Throughput	maximum	22637.87	22637.87	17342.36	17342.36	17823.60	17823.60
Throughput	minimum	229.51	0.51	228.33	0.67	87.19	1.36
Throughput	average	4017.11	13.76	4310.72	14.01	4672.81	15.38
Throughput	average (KBps)	3.92		4.21		4.56	

	Scenario	IV		V		VI	
	pause time	1 sec		2 sec		5 sec	
	maximum speed	1 m/s					
		Bytes	*packets*	*Bytes*	*packets*	*Bytes*	*packets*
Routing	average routing packet size	286.1		279.25		278.65	
Routing	sent	12622368	44119	12778572	45760	12812964	45982
Routing	overhead	136.34%	146.02%	131.09%	146.49%	135.49%	151.38%
		seconds		*seconds*		*seconds*	
Delay	maximum	37.70564		43.71765		45.86712	
Delay	minimum	0.00194		0.00194		0.00194	
Delay	average	0.70457		0.72968		0.73382	
Throughput		*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*
Throughput	maximum	21714.75	21714.75	19243.40	19243.40	16165.40	16165.40
Throughput	minimum	13.34	0.33	121.19	1.20	125.01	2.61
Throughput	average	3558.01	12.21	3919.73	12.73	4415.64	15.67
Throughput	average (KBps)	3.47		3.83		4.31	

D.2.4 Network of 50 Nodes Using AODV

	Scenario	I		II		III	
	pause time	0.5 seconds					
	maximum speed	1 m/s		2 m/s		5 m/s	
		Bytes	*packets*	*Bytes*	*packets*	*Bytes*	*packets*
Routing	average routing packet size	135.81		152.11		143.3	
	sent	8587736	63235	8365224	54996	8346368	58244
	overhead	53.51%	238.06%	61.31%	218.85%	34.13%	179.12%
		seconds		*seconds*		*seconds*	
Delay	maximum	18.20517		10.21601		8.60545	
	minimum	0.00194		0.00194		0.00194	
	average	0.62147		0.74470		0.64232	
Throughput		*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*
	maximum	23004.12	23004.12	13917.75	13917.75	20277.47	20277.47
	minimum	31.38	0.10	28.11	0.47	145.34	0.45
	average	3981.82	12.24	4407.91	13.10	4679.46	14.42
	average (KBps)	3.89		4.30		4.57	

	Scenario	IV		V		VI	
	pause time	1 sec		2 sec		5 sec	
	maximum speed	1 m/s					
		Bytes	*packets*	*Bytes*	*packets*	*Bytes*	*packets*
Routing	average routing packet size	149.72		146.29		165.91	
	sent	8385344	56006	8452360	57779	8116664	48923
	overhead	65.74%	231.04%	49.06%	203.92%	67.05%	200.81%
		seconds		*seconds*		*seconds*	
Delay	maximum	8.72190		17.94677		9.67683	
	minimum	0.00194		0.00194		0.00194	
	average	0.68107		0.68509		0.81271	
Throughput		*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*	*Bytes/sec*	*pckts/sec*
	maximum	21848.60	21848.60	21306.83	21306.83	17349.44	17349.44
	minimum	110.43	1.84	70.78	0.65	192.21	0.39
	average	3819.44	11.23	4120.86	12.58	3627.85	10.82
	average (KBps)	3.73		4.02		3.54	

Appendix E:
Current WiFi-enabled Mobile Phones

Following is a list of the WiFi-enabled mobile phones available in the market as of May 13th, 2006.

BenQ P50
Benq-Siemens P51
Eten M600
HP iPAQ h6315 / 6310 / 6320 / 6325 / 6300
HP iPAQ hw6925 / 6920 / 6900
HP rw6800 / Quanta O2 Xda Atom
HTC Apache / PPC-6700 / XV6700
HTC Blue Angel / SX66 / MDA III / PDA2k
HTC Prophet / i-mate JAMin / Qtek S200
HTC Tornado / T-Mobile SDA / i-Mate SP5m / Orange C600
HTC Universal / JASJAR / EXEC
HTC Wizard / T-Mobile MDA / i-mate K-JAM / Cingular 8125 / 8100
Motorola CN620
Motorola M1000
Motorola MPx
Nokia 6136
Nokia 9300i
Nokia 9500 Communicator
Nokia E60
Nokia E61
Nokia E70
Nokia N80
Nokia N91
Nokia N92
Nokia N93
Panasonic Toughbook CF-P1
Samsung SCH-i730
Samsung SGH-i750
Sony Ericsson P990