UNIVERSITÉ LIBRE DE BRUXELLES
Faculté des Sciences Appliquées

Année académique 2005-2006

Design and Implementation of a Physical-Constrained Wireless Routing Algorithm in Linux

Coordinateur de mémoire:
Esteban Zimányi
Sus-coordinateur de mémoire:
Jean-Michel Dricot

Mémoire de fin d´études présenté par
Pedro Miguel Antunes
en vue de la obtention du grade
"Engenheiro Informático"
a l´Université de Coimbra sous un accord
Sócrates-Erasmus

Contents

1 Acknowledgments

To all whom I care for and particularly to those who care for me...

I would like to take this opportunity to thank all who have helped me through this stage in my life, always supporting me in the worst times and sharing my joy in the best.

To my parents, Manuel and Maria, who have taught me the true values of honesty and perseverance, and always saught to point out to me the right path in life. Thank you for never having given up on me and for always having believed in my potential. To all of my friends in Coimbra, who always stood by me, even in the darkest times, and showed love and friendship beyond all boundaries. To my best friends Tony, who showed me the beauty behind a computer and taught me most of what I know now, not only in "101001 land" but also in life, and Ricardo, whom I have known for a long time now, and grew up with, sharing knowledge, experience and life's teachings. To João Seabra, who has given me such good advice for so long and has taught me the good lessons of perseverance, and also the value of being calm and serene. To JP and DOT for their technical advice and expertise, always helping me regardles of all difficulties. To the friends that I have made here in Brussels, Wolfgang, Mark and Elisa, who helped me to realize that we are never alone in this world.

To my mentor in Portugal, Mário Rela, who has always been available to help me, with endless enthusiasm and interest in my work. To my coordinator in Brussels, Jean-Michel Dricot, who has always received me with a smile and helped me solve the problems arising from my work. Finally, to the department director, Prof. Esteban Zimanyi, who took me into his prestigious team, although I was just a beginner and needed so much guidance. To Jean Tourrilhes, who helped me to get my work started and pointed me in the right direction, and Luke Klein-Berndt of the NIST in whose work I based my project. Last but definitely not least, to the one who will always be the true love of my life, Carrie Tyler, who has so patiently spent the last three years of her life making me endlessly happy, neglecting my long list of defects and my world famous impatience.

To all of you... My life!

"If a man does his best, what else is there?" - General George S. Patton (1885-1945)

2 Abstract

Nowadays there are many routing protocols and algorithms widely available for use in adhoc networks, but none of them was ever specifically designed for wireless networks.

In wired structures, signal integrity is taken for granted and distance is not relevant (to a given point) due to very low transmission latencies. This creates a problem for the current routing world scenario: how should we deal with routing problems in wireless networks, where packet loss can result from endless factors leading to bandwidth degradation?

To try and solve this problem, we are going to develop a routing application based on an algorithm presently being developed in a cooperation between Universitá di Parma (UP) and Université Libre de Bruxelles (ULB). This routing algorithm is called *"Modified Ad-Hoc On-Demand Distance Vector"* [1] and was first idealized in the UP, in Italy, by Professor Gianluigi Ferrari, a brilliant young professor who has a very strong interest in ad-hoc wireless networking and is a specialist in bit error rate in radio communications and signal theory, and is being widely perfected by Jean-Michel Dricot, a promising assistant professor of the ULB, in his doctoral thesis.

Through this thesis we will demonstrate how this algorithm works and which are the advantages in comparison with a similar generic algorithm, explain all the background behind the algorithms philosophy and plan and execute a full software project to implement the suggested protocol based on [2].

At the end of this document we will show that there are concrete advantages to this concept, through some limited testing and experimentation, and how it could be applied in the real world and marketed as a proper, commercially viable, solution.

3 Acronyms

AES - Advanced Encryption Standard

AP - Access Point

API - Application Programming Interface

AODV - Ad-hoc On-demand Distance Vector

AWGN - Additive White Gaussian Noise

BER - Bit Error Rate

BPSK - Binary Phase Shift Keying

BSS - Basic Service Set

DHCP - Dynamic Host Configuration Protocol

DSDV - Destination-Sequenced Distance Vector

EAP - Extensible Authentication Protocol

FTP - File Transfer Protocol

IEEE - Institute of Electrical and Electronics Engineers

IOCTL - I/O Control (allows an application to control or communicate with a device driver outside the usual read/write of data)

LAN - Local Arean Network

MAODV - Modified Ad-hoc On-demand Distance Vector

MAODV-PC - Modified Ad-hoc On-demand Distance Vector with Power Control

NIST - National Institute of Standards and Technology

PDR - Packet Delivery Ratio

PEAP - Protected Extensible Authentication Protocol

POE - Power Over Ethernet

QOS - Quality Of Service

RERR - Route Error message

RFI - Radio Frequency Interference

RREQ - Route Request message

RREP - Route Reply message

RTT - Round Trip Time

SNR - Signal-to-Noise Ratio

TKIP - Temporal Key Integrity

TTLS - Tunneled Transport Layer Security

VoIP - Voice over Internet Protocol

WE - Wireless Extensions
WEP - Wired Equivalent Privacy
WLAN - Wireless Local Area Network
WPA - WiFi Protected Access
WT - Wireless Tools

4 Introduction

Wireless networks, in the most known form of IEEE802.11a/b/g, have found a great development in the latest years.

The technology has undergone big enhancements, its capabilities, functionalities and performance have expanded greatly. These enhancements have brought several improvements in security mechanisms, available bandwidth and transmission power which in turn produce safer and more reliable data communication, higher transference rates, lower latencies and higher reachable radius.

There are two main kinds of structural methodologies in wireless networks. They can be set up in infra-structure mode (Access Point mode), or in a peer-to-peer mode called Ad Hoc networking.

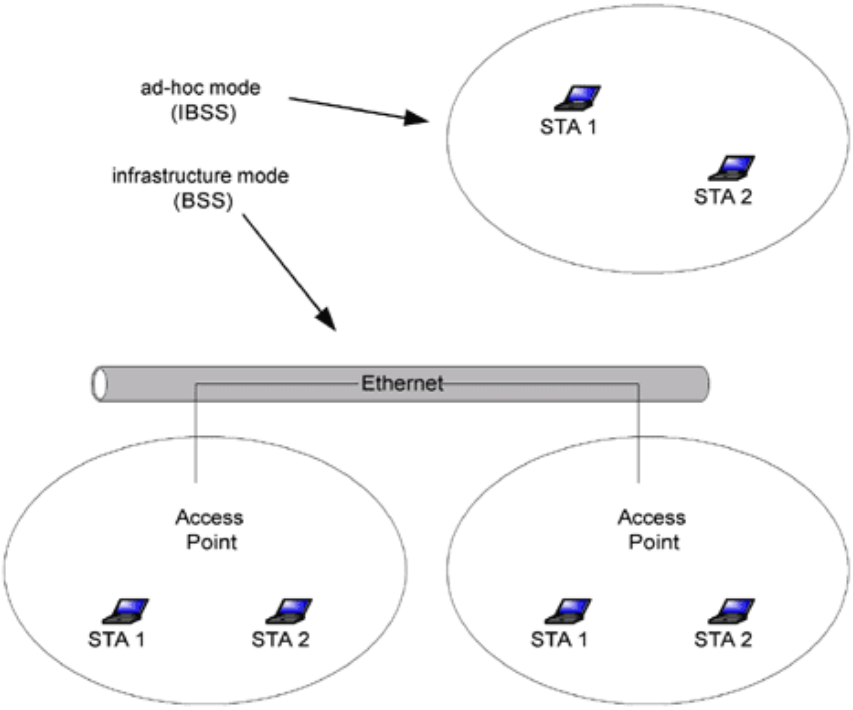

Figure 1: Ad-hoc VS Structured [3]

One of the most spoken paradigms in wireless networks is the ad hoc model. It basically represents a direct communication mode between terminals (nodes), without the need for a central point of convergence.

Unfortunately ad-hoc networks still face several problems that have been studied in research centers around the world. The current unavailability of an appropriate routing protocol, bandwidth reduction along node hops, synchronization in the physical layer (medium lock) and electro-magnetic interference from surrounding wireless signals (including neighbour nodes in the same network) among others.

This thesis will focus on the high scale capabilities of the ad-hoc model, its strengths and weaknesses, possible solutions for these problems. It also includes a number of real life scenarios/cases where it can be applied, but the study will mainly focus on the search for the appropriate routing protocol for this kind of topology and some performance enhancement mechanisms related to it.

4.1 Current World Wide Scenario

Ad hoc network technology can be considered to be one of the next steps in the evolution of wireless architecture, promising greater flexibility, reliability and performance over conventional Wi-Fi wireless LANs (WLANs).

Unlike WLANs that have a wired connection between access points and switches or a wired LAN, in wireless ad hoc networks, all communication among network nodes, and between nodes and clients, is through a radio link instead of cable. There are already some vendors that propose some ad hoc solutions and additionally support connectivity between their networks and an enterprise network via either radio (802.11a/b/g) or direct Ethernet (100BaseT) cable connection for users or service providers who want to connect the two networks.

The major benefit of this emerging technology is that it enables anyone with a wireless adapter card in their laptop, tablet or PDA to communicate on a peer-to-peer basis, in addition to accessing a WLAN. The key advantages of wireless ad hoc networks include ease of installation, no cable costs, automatic connection among all nodes, network flexibility, automatic discovery of newly added nodes, redundancy and resilience. According to a recent study by ABI Research, the use of ad hoc networking technology for large-scale wireless deployments could easily grow by a factor of 10 over the next five years [4] [5] [6].

Wireless ad hoc networks can be used either outdoors or indoors, in the home or small office. The deployment of outdoor networks for a metropolitan area, such as citywide or community-wide "hot zones," or large "hot spots," e.g., corporate or university campuses,

will likely contribute to the anticipated substantial market growth.

Wireless ad hoc network vendors/providers are growing in number and now include some network communication companies that have been created to deal specifically with design and deployment of ad hoc networks, including Co Co Communications, BelAir Networks, Firetide, Mesh Dynamics, PacketHop, Sky Pilot, Strix Systems and Tropos Networks, each of which provides its own hardware/software solution.

In addition, legacy network vendors that currently offer ad hoc network solution products include Motorola MeshNetworks, Nortel, and most recently, Cisco Systems. The entrance of Cisco into ad hoc network world further prove the industry and bolsters growth predictions. There are two basic types of ad hoc/mesh architectures [21]: fixed and mobile. Fixed ad hoc networks contain network nodes deployed throughout an indoor or outdoor area or zone specified for wireless connectivity. Some ad hoc solution vendors support both indoor and outdoor fixed systems with seamless connection between the two.

In contrast, mobile networks are tuned for high mobility and enable the creation of autonomous mobile networks composed of off-the-shelf devices, each containing an 802.11 wireless radio. Mobile ad hoc networks can be deployed instantaneously anywhere and provide broadband coverage wherever users are. Mobile networks grow in capacity, range and redundant connections as more users join. Additional client nodes fill in gaps in the wireless coverage.

Mobile networks have been used by the American military to establish temporary networks under battlefield conditions. They are also applicable to ad hoc applications by first responders, such as police, fire-fighters and EMS crews because they can be immediately established and then terminated when a mission is complete.

4.2 The Strengths

Ad hoc technology enables new applications and offers several benefits over conventional WLANs. An ad hoc network is easier to install and is less costly than wired networks, since it uses radio signals in place of cable to connect nodes.

Resilience is a key attribute of ad hoc networks. A mesh network supports multiple paths between network nodes, hence there is no single point of failure, and nodes can be added to increase redundancy. Ad hoc networks can load-balance traffic to prevent congestion and reroute traffic to bypass congested nodes. An ad hoc network distributes

intelligence along the network instead of centralizing it. These networks can also reduce or eliminate T1, DSL and other service costs for backhaul/backbone traffic.

Another key benefit of this kind of topology is its ability to circumvent large objects, such as buildings, that block the signal path; signals are simply rerouted through alternate nodes that bypass the building or object. Nodes can also be added to cover dead zones or to compensate for blocked signal paths.

4.3 The Weaknesses

Ad hoc networks also have weaknesses. They can be affected by bandwidth depreciation, radio interference and per-hop latency as networks grow. However, increasing the size of a mesh network does not automatically cause performance to decline. In some cases, performance is actually improved by the added capacity (bigger size), providing the network with load balancing due to the existence of more alternative routes.

Bandwidth decline and latency occur when traffic travels between the network nodes [7] [8]. For each hop, bandwidth is cut in half (bandwidth = 1/n where n = the number of hops) and latency increases (each effect occurs for technical reasons explained below). For example, a hop count of four would reduce throughput to 25 percent.

A worst-case scenario-and also, unfortunately, the most realistic one-for bandwidth degradation is where the bandwidth degrades by $\frac{1}{2(n-1)}$ being n the number of nodes the packet travels between nodes. This is the most realistic case because in most mesh networks, each node is surrounded by several neighbouring nodes that cause radio interference. In this case, throughput would be reduced to 13 percent for a hop count of four. So the key to improve performance is sound network planning-at least in fixed networks, where such control can be fully exercised. Network planners must try to minimize excessive hop counts and to balance loads.

Radio frequency interference (RFI) can considerably affect the devices in a wireless ad hoc network, causing transmission errors and deteriorated network performance. RFI can be radiated from a whole gamut of sources, e.g., other electronic devices, cordless phones, wireless security cameras, microwave ovens, computers, etc. RFI can be continuous, temporary or intermittent, depending on RFI source activity.

Many wireless devices have pluri-directional antennas, which extend wireless connectivity but also can be another limiting factor. Since pluri-directional antennas propagate

a signal in a 360 degree pattern, the network suffers from RF interference and collisions as the number of users and nodal transmitters on the network increase. An omni-directional antenna also limits the coverage range of each node, because its energy is dispersed equally in all directions instead of being focused in one direction. In contrast, directional antennas, as the name suggests, concentrate all their energy in one direction. High gain antennas extend signal range, allowing nodes to be placed further apart to require fewer nodes within a network and to reduce power consumption, helping to achieve the network planning goals described above.

Fixed ad hoc solution vendors include one or more antennas in their network nodes. Some [13] provide antenna selection to switch the signal path in the appropriate direction to eliminate the need for the installer to manually point the antennas. The use of multiple frequencies and channels within a ad hoc network is also used to preclude interference [14]. Firetide's MeshBridge feature does just that for each ad hoc segment, to optimize a segment to local operating conditions.

Network latency and jitter affect wireless ad hoc networks that support data, voice, and video traffic. Voice and video performance drops as latency increases and jitter becomes extensive, so hop counts must be kept down in order to minimize latency.

4.4 Scalability and Performance

The scalability and performance of ad hoc networks rely heavily on the network's architecture to overcome the limitations previously mentioned. For example, early-generation ad hoc networks had limited scalability and suffered performance degradation over multiple hops. That is because each node contains a single radio, operates in half-duplex mode and shares the same radio frequency. This is called a Basic Service Set (BSS).

All radios on the same channel must remain silent until the packet completes its total hops within the same BSS. Each sequential hop reduces bandwidth by one-half of that at the previous node, because the node must handle the packet twice-receiving it and then sending it on. In addition, waiting for a packet to complete its total number of hops introduces substantial delay; latency of up to 50 milliseconds per hop can be introduced, which at four hops is 200 milliseconds. This much delay precludes delay-intolerant applications such as voice and real-time video.

As noted above, excessive hop counts can be minimized through good network archi-

tecture design.

While single-radio networks tend to be the least expensive, scalability and performance can be greatly improved via more-sophisticated (and more expensive) multiple-radio ad hoc networks. This architecture provides a "backbone," called the backhaul network, which interconnects all nodes and handles traffic between nodes. Separate radios in each node, tuned to different frequencies, reduce backhaul interference and congestion. Client access is supported by separate 802.11a/b/g links provided by each node. Vendors refer to this type of network as a structured mesh network, since it differs from "unstructured," ad hoc mesh networks.

In two-radio mesh networks, each node contains one radio for backhaul, while a second radio acts as an Access Point (AP). In other words, one of the node's radios "talks" to the client, and the other radio "talks" to peer nodes.

Although the dual-radio architecture improves performance and scalability, even greater performance and scalability can be delivered with a more complex multi-radio architecture, which uses three or more radios per node. A three-radio node adds a second radio for the backhaul portion of the network; having two radios per node for backhaul enables a node to operate in full-duplex mode, enabling it to transmit and receive backhaul traffic concurrently.

In practice, ad hoc network architecture may combine single, dual, and multi-radio nodes within a network to address specific network requirements and to maintain the lowest cost. As single-radio network segments have limited scalability, they are limited to fewer nodes to reduce hop counts.

Some of the multi-radio products [15](i.e., three or more radios) are configured with one 802.11b/g radio for client access and two 802.11a radios for backhaul. Others [6] configure their network with up to six radios per node. Core nodes contain four to six radios, while access nodes contain two or four radios, and edge nodes have one or two radios since they do not relay any traffic.

Large-scale ad hoc/mesh networks provide metropolitan coverage. A vendor [16] has deployed its network for continuous coverage in several U.S. cities, including Oklahoma City, Corpus Christi and Chaska. In Tempe, AZ a new one has been deployed [6], that supports voice/video/data operation over a 100 square km area. The largest ad hoc metropolitan network in the world, with 10,000 Access Points covering 270 square kilometers, is in Taipei, the capital of Taiwan [17].

4.5 Network Reliability

High reliability is the hallmark of wireless mesh networks due to their fault-tolerant architecture. Self-healing fail over capabilities automatically sense a path outage and reroute traffic. Some network vendors provide backup radios and antennas for backhaul and access portions of the network for even greater reliability.

Ad hoc networks also can support traffic load balancing over multiple backhaul links, in response to increasingly heavy traffic. Power supply redundancy and battery power backup are also required for maximum resilience.

Networks designed for outdoor use provide rugged construction and weatherproof housing, in addition to features such as lightning protection, electrostatic discharge protection and internal heaters for low-temperature operation in cold environments.

4.6 Network Security

Ad hoc networks provide the same security features and mechanisms as WiFi WLANs. Backhaul networks must have the greatest protection and are typically encrypted, because they support all traffic for the entire network. SkyPilot, for example, provides 128-bit AES (Advanced Encryption Standard) in each wireless node in its backhaul network and uses MD-5 certificates to ensure authorization.

Network vendors support standard client authorization measures in their Wi-Fi access points including 802.1x and RADIUS servers. Vendors also support a variety of encryption techniques, such as WEP, WPA, EAP-TTLS, EAP-PEAP, TKIP (Temporal Key Integrity), 802.11i etc., to support mobile customer requirements.

4.7 Standards and Interoperability

Proprietary ad hoc solutions provide standard-based network access using 802.11 on the network access radios within the AP, to communicate with users' 802.11 cards in laptops and hand held devices. Most networks support 802.11b/g. However, because commercially-available wireless mesh networks are relatively new, there is no IEEE-based standard yet for creating the mesh between nodes.

Today's enterprise ad hoc networks are based on vendor-proprietary mesh algorithms and routing protocols and do not support interoperability between competitive products. More than 70 protocols have been introduced for routing data across mesh networks.

The IEEE 802.11 Working Group appointed Task Group S (TGs) to develop a world wide ad hoc/mesh network standard. At the IEEE 802.11 meeting in San Francisco in November 2005, 15 proposals were submitted for a Wireless Mesh Network standard. TGs hopes to have a completed draft standard in 12 to 18 months, which means the 802.11 s standard could be ratified before 2008 if all goes smoothly.

Interoperability will require thorough testing and certification by a consortium such as the WiFi Forum.

5 Theoretical Basis - Routing Protocols and Main Concerns

One of the leading routing protocols for ad hoc networks is the Ad hoc On-demand Distance Vector (AODV) routing protocol. It is based on temporary routing tables which contain the specific path cost metric (number of hops) to its destinations.

Previous studies have shown that although this protocol is easily applicable in any virtually error-free environment (high speed fiber optics), its performance tends to drop in high speed node connections and high active node number scenarios for wireless ad hoc networks [9]. Part of the results obtained suggest that other variables should be taken in consideration, specially important physical layer characteristics like the bit error rate (BER) at the end of the multi-hop route that would give a good representation of the physical medium status, like traffic congestion and signal quality of the existing routes.

A direct derivation of this protocol as been developed in order to approximately minimize BER at the end of a multi hop route. It has been called Modified AODV (MAODV). Studies and simulations with Network Simulator 2 (NS-2) have shown [1] that MAODV generally guarantees a worse performance value than the AODV, as often the path with the lowest number of hops is not the chosen one, but instead the path represented with the shortest distance between nodes (shortest possible longest hop of the route).

Although this could seem like a bad strategy for the election of a protocol of choice, as it will largely increase the packet transmission delay (latency), we show that combining this routing method with a good transmission power control mechanism (PC) dramatically increases the delivered packet percentage. As such, the proposed routing protocol allows the reduction of the average transmission power of the nodes (as closer nodes are the ones chosen as next hops for the selected routes). The results suggest that MAODV with transmission power control (MAODV-PC) is the preferred protocol for use in indoor ad hoc networks.

5.1 Background: Ad hoc On-demand Distance Vector (AODV)

VERY IMPORTANT NOTE: This chapter is taken from "Ad-hoc On-Demand Distance Vector Routing" by Charles E. Perkins and Elizabeth M. Royer. It was very important to preserve the originality of the document, for obvi-

ous reasons, so only minor changes have been made in the contents of the original text. [10]

5.1.1 Protocol Functioning

The ad hoc on-demand distance vector routing (AODV) is a reactive routing protocol that uses temporary routing tables with one entry per destination. It represents a cooperative engagement of a collection of mobile or fixed wireless nodes without the required intervention of any centralized point of convergence or existing infrastructure. AODV provides loop-free routes even when performing the task of rebuilding broken routes. As it is not based on regular route advertisements, like normal distance vector protocols, bandwidth requirements and bandwidth use is substantially smaller then the protocols that demand such types of advertisements. In any case, it is still possible to maintain the major advantages provided by the basic types of distance vector routing protocols.

A different variation of this kind of protocol has been proposed; the denominated Destination-Sequenced Distance Vector algorithms (DSDV) [18]. These are effective for designing ad hoc networks for small node populations. However, because they depend on regular broadcasts of the routing tables and have to maintain at all times a complete list of all of the available routes, the routing network traffic overhead has a tremendous negative effect on big network architectures with a high node population. However, maintaining a complete list of all of the routes does reduce the latency related to the route acquirement procedure, meaning lower transmission times for the first packet to a specific destination while also meaning higher bandwidth needs for the periodical routing table broadcast. In any case, the on-demand protocols are systems which have been designed to overcome some of these issues. The capability of inquiring neighbouring nodes about information on the routes which have not been settled or have expired only when required is beneficial to overall bandwidth requirements. Routing overhead is much less demanding in these cases than with the periodical advertisement distance vector protocols. Such systems must take precautions in order to limit the time spent in acquiring new routes, otherwise network nodes might experience unacceptably long waiting periods in order to be able to transmit urgent information. Applicational protocols as VoIP or video transmission feeds will suffer tremendously from these latency increases, as small transmission times are normally more crucial than higher bandwidth rates.

Although the development of the AODV protocol has been largely motivated by the need for routing schemes for limited range broadcast media such as those utilized by infrared or radio frequency wireless communication adapters, this protocol never intended to use specific characteristics of the physical medium, nor to handle specific problems posed by channeling needs of radio frequency transmitters. Nodes that use multi-channel transmissions are presumed to be able to do so, without the acknowledgment or interference of the routing protocol. The algorithm works in wired media as well as wireless environments, as long as there are available links through which the packets can be transmitted. The only requirement for the physical medium is that neighbouring nodes must be able to detect each others broadcasted transmissions.

There is possibly no symmetric link between two nodes, or in other terms, the route is not established in both directions and one node can hear the other but the second one cannot hear the first. The AODV protocol was not designed to make use of such links, but future enhancement plans predict the development of a mechanism that would take advantage of such connections.

5.1.2 Protocol Algorithm

In the so called pure on-demand route acquisition systems, nodes that do not lie on active paths do not maintain any routing information and do not take action in any kind of periodical route advertisement. In fact, a node does not contain the information of a path to another node and will not try to acquire this information unless it needs to communicate with another given node or to act as an intermediate forwarding station for a routed packet.

When a local connection has to be established for packet routing, a node becomes aware of the existence of other nodes in the surrounding area by applying a series of techniques that include broadcasting a local hello message (never in a full network coverage broadcast). As the routing tables in each neighbouring node are organized in order to reduce the path search time as well as the response latency value, the response time is optimized to reply quickly to these requests so that new routes are acquired.

Amongst the algorithm's objectives, we can consider the following the most important:

- To broadcast discovery packets only when necessary.

- To distinguish between local connectivity management (neighborhood detection) and general topology maintenance.
- To disseminate information about changes in local connectivity to those neighboring mobile nodes that are likely to need the information.

The AODV protocol uses a broadcast based route discovery mechanism [19]. It dynamically establishes the routing table entries at intermediate nodes and to maintain the most updated entries, the protocol uses a destination sequence numbering method that allows each node to maintain a monotonically increasing sequence number counter which is used to supersede stale cached routes. The combination of these techniques provide an algorithm that assures a more efficient bandwidth usage by minimizing the network usage for control and data traffic, and which is capable of responding and adapting to network topology changes while assuring loop-free routes.

5.1.3 Route Discovery

The route discovery process is initiated when a source node has to communicate with another given node for which no predetermined path exits in its routing table. The nodes have to maintain two counters: a node sequence counter and a broadcast id. If a node wants to discover a new path across the network, it broadcasts a route request (RREQ) packet to all of the surrounding nodes. This packet contains the following fields:

- source address
- source sequence number
- broadcast id
- destination address
- destination sequence number
- hop count

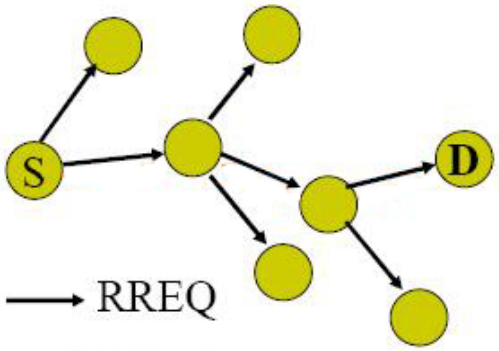

Figure 2: RREQ propagation scheme

The sole purpose of the parameters source address and broadcast id is to identify the RREQ packet. The broadcast id is a counter that keeps track of the number of RREQ packets issued; its value is increased for every new one that is transmitted by the source node. Each neighbouring node either contains the required path in their table, in which case it replies by sending a route reply packet (RREP) back to the source, or re-broadcasts the RREQ packet to its own neighbours after increasing the hope count value. It is important to acknowledge that a node might receive several copies of the same route broadcast packet from the various surrounding nodes. If an intermediary node receives a RREQ packet with repeated source address and broadcast id fields it drops the redundant packet and does not broadcast it again. If a node cannot provide a route reply, it keeps a series of parameters in order to be able to build a reverse path setup [19], along with a forward path setup that will be transmitted together with an eventual RREP packet. These parameters are the following:

- Destination IP Address
- Source IP Address
- Broadcast id
- Expiration time for reverse path route entry
- Source nodes sequence number

5.1.4 Reverse Path Setup

For the reverse path setup to occur there are two essential sequence numbers (besides the broadcast id) that are included in a RREQ packet: the source sequence number and the last destination sequence number known to the source node. The source sequence number is used to assure freshness information about the reverse path to the source, and the destination sequence number specifies how fresh a route to a destination node has to be before it will be acknowledged by the source node.

A reverse route is automatically built from all the nodes back to the source while the RREQ packet travels through the network from the source to its various destinations. To build this path, each node creates a record of the address of the neighbouring node from which it received the first copy of the RREQ packet. These records should be kept by each node long enough for the RREQ packet to travel the whole network and produce a reply to the original sender.

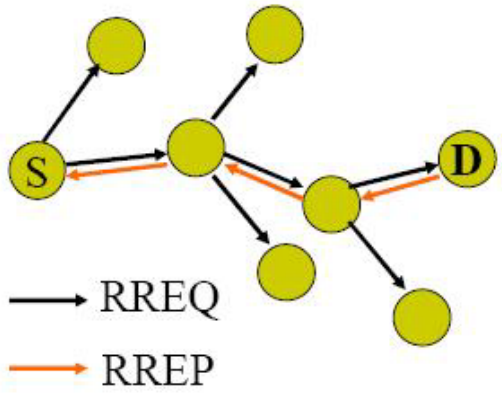

Figure 3: Reverse path construction scheme

5.1.5 Forward Path Setup

When a RREQ arrives at a node that has the information for a needed route, maybe even for the destination itself, it will be first checked by the node to see if it was received over a bi-directional link. If an intermediary node has the information for the required route, it will assess whether the path is "fresh" by comparing the destination sequence number

in its own table to the destination sequence number in the RREQ. If the RREQ' sequence number for the destination node is higher than the one present in the intermediate node's table, the intermediate cannot use its own path entry to reply to the RREQ. Instead, it must re-broadcast the RREQ as if there were no path in its routing table. The intermediary node will only be able to reply when the specified route present in its table has a sequence number greater or equal to the one included in the RREQ packet. Should it have an updated route in its table and if the RREQ packet has never been previously processed, the intermediate node sends a Route Reply Packet (RREP) back to the neighbour from which it received the original RREQ. The RREP packet contains the following fields:

- Source address
- Destination address
- Destination sequence number
- Hop count
- Lifetime

When a broadcasted packet arrives at a node which can provide a current path to its destination, a reverse path has been established to the source node of the RREQ. As the RREP propagates back to its source, each intermediate node registers a forward pointer to the node from which the RREP was transmitted, updates its timeout information for the correspondent route on its table both for source and destination, and records the latest/the most recent sequence number for the requested destination. Nodes that are not in the path used by the RREP will suffer a timeout after ACTIVE_ROUTE_TIMEOUT (normally 3000 msec) and will drop the reverse pointers to the source.

Every intermediate node that receives a RREP propagates the first packet of this kind for a given source toward s that source. If it receives more RREP packets, it will update its routing information and propagate the RREP only in case its sequence number is greater than the one present in the last received RREP, or if the sequence number is the same but the RREP contains a smaller hop count. All other received RREP packets should be dropped by the node. This will decrease the number of superfluous RREP packets

traveling through the network towards the source, therefore allowing a better bandwidth management and lower routing overhead for the procedure while also assuring that the routing information that prevails is always the one that is most up-to-date and represents the quickest path. The original source node will be able to initiate data transmission as soon as the first RREP is received, and will update its routing information in case a new RREP containing a better path is received.

5.1.6 Routing table management

Besides the source and destination sequence numbers that all nodes should maintain, more important routing information can be found in the nodes route table entries, called the soft-state information. Every registered reverse path has an associated timer called the route request expiration timer. This timer has the single purpose of dropping the reverse path routes from the nodes that do not lie on the path from the source to the destination. This expiration time varies depending on the size of the whole ad-hoc network. There is also another important parameter that all nodes should maintain in their routing entries, which is called the route caching timeout, which represents the amount of time after which a given route will cease to be considered valid.

In each routing table entry there is also the address of the active neighbour nodes for which a packet should be transmitted to be delivered to a given destination node (next hop node). A neighbouring node is considered active (for that given destination) if it delivers or relays at least one packet for that destination within the most recent active timeout period. This information is maintained so that all active source nodes may be notified when a link along a path to the destination breaks. A route entry in the table is considered active if it is being used by any active neighbour, and the path from a source to a destination node, which is followed by packets along active route entries, is called an active path. Keep in mind that all routes in the node routing table are connected to a destination sequence number, which guarantees that no loops form on the path, even under delicate conditions like out-of-order packet delivery and high node mobility.

A node maintains a route table entry for each relevant destination, which contains the following information:

- Destination node address

- Next hop node
- Number of nodes to destination (metric)
- Sequence number for the destination
- Active neighbors for this route
- Expiration time for the route table entry

Every time a route entry is used for data transmission from a source node to a give destination, the expiration timeout for the entry is reset to the current system time plus the active route timeout parameter value.

If a new route is discovered by a node, it will compare the destination sequence number of the new route with its own register of this sequence number for the current node. The route with the largest sequence number is the chosen one, and if the sequence numbers have the same value, then the prevailing route will be the one with the smallest number of hops (lowest metric) to its destination.

Table 1: Route table example (Part 1)

IP address	netmask	sequence number	number of hops	distance to next hop
10.0.0.7	255.255.255.0	13	4	62
10.0.0.2	255.255.255.0	18	1	54
10.0.0.5	255.255.255.0	7	6	92
10.0.0.4	255.255.255.0	12	2	71
10.0.0.1	255.255.255.0	23	7	92

Table 2: Route table example (Part 2)

IP address of next hop	rreq id	lifetime(ms)	net device to use	is route valid?	is it self route?
10.0.0.3	17	2700	wlan0	1	0
10.0.0.5	22	1450	wlan1	1	0
10.0.0.2	8	700	wlan0	1	0
10.0.0.1	2	1900	wlan1	1	0
10.0.0.2	15	2200	wlan0	1	0

5.1.7 New route acquirement

The mobility of the nodes that do not lie along an active path will not affect routing towards the path's destination. If the source node changes location during a protocol information exchange, it should re-initiate the information exchange in order to discover a new viable route to the packet's destination. When the destination node or an intermediate node moves, a special RREP is sent to all affected source nodes. The periodical Hello message transmission assures that only symmetrical links are maintained and that link failures between neighbouring nodes are correctly detected. As an alternative to this method, to assure lower latency values, it has been suggested that these failures be detected by using link-layer acknowledgements (LLACKS). A link failure can also be detected if the attempt to forward a packet to the next hop fails.

Once a "next hop" node becomes unreachable, the node upstream from the break propagates an unsolicited RREP with a newly generated sequence number (greater than the previously known one) and hop count of infinite to all active upstream neighbours which relay that message to their own neighbours and so on. The described process continues until all active source nodes on the path receive this package, ending because AODV maintains only loop-free routes and because there will always be a finite number of nodes in the ad hoc network.

Upon receiving the RREP identifying a broken route, the source nodes can start inquiring for new routes to their destination if still needed. To assess if a route is still required for packet transmission, the node can see if the route has been used recently (by consulting the value of the timeout), as well as inquiring the upper layer protocol control blocks to see whether the connection remains open using the indicated destination. If the source node (or any other node known to be in the previous route path) decides that it still needs to rebuild the route to the destination, it sends a RREQ with a greater destination sequence

number (normally a single digit)than the previous know route. This ensures that it builds a new, viable route, and that the nodes which still consider the broken route as valid will not reply to its request.

5.1.8 Neighborhood discovery and management

The neighbouring nodes can be discovered by one of two means. Whenever a node receives a broadcast from a neighbouring node, it updates the available local connectivity information that it contains in order to ensure that this particular neighbour is included. If a node did not have the need to transmit or relay any packets within the hello interval time value, it will broadcast a hello message to all of its neighbours, which is nothing more than a RREP containing its identity and sequence number. A node's sequence number is not incremented upon the transmission of a hello message.

This message is prevented from being transmitted outside of the neighbourhood of the node, because it contains a time to live (TTL) of 1. All neighbouring nodes that receive the packet update their local connectivity information to the node in question. Receiving a broadcast or a hello message from an unknown neighbour, or failing to receive a number (stipulated by the allowed hello loss parameter) of consecutive hello messages from an active node that was previously in the neighbourhood, is a good indicator that the neighbourhood's structure has changed. Nonetheless, keep in mind that not receiving these messages from non-active nodes does not trigger any protocol reaction. If hello messages are not received from the next hop along an active path, the active neighbours using that next hop will be sent a notification of the link failure as described in the previous chapter. The optimal value for the allowed hello loss is two, as described by the author of the protocol [10].

This local connectivity management technique (with hello messages) is also used to ensure that only symmetrical links are used and that nodes that do not share bidirectional links are not considered neighbours. For this purpose, each hello message sent by a node contains a list of the nodes from which it has heard. Each node checks to make sure that it only uses routes to neighbours that have heard its hello message. To better manage the available local bandwidth, such checking should only be performed if explicitly configured into the nodes.

5.2 Modified Ad hoc On-demand Distance Vector

VERY IMPORTANT NOTE: This chapter is based on "Physical Layer-Constrained Routing in Ad-hoc Wireless Networks: A Modified AODV Protocol with Power Control" by Gianluigi Ferrari, Simone A. Malvassori, Marco Bragalini and Ozan K. Tonguz. It was very important to preserve the originality of the document, for obvious reasons, so only minor changes have been made in the contents of the original article by these authors. [1]

In May 2005, at the International Workshop on Wireless Ad-hoc Networks (IWWAN'05) meeting, in London, an Italian university professor called Gianluigi Ferrari presented his work on a new AODV protocol that he had developed, called Modified AODV with Power Control. This protocol showed interesting aspects that varied quite a bit from the regular AODV algorithms, specially the usage of a different metric other than the hop count.

The MAODV is a direct derivation of the AODV protocol [10]. It characterizes itself by choosing a different route than the one that would be chosen by the AODV. It leads to the selection of the route that minimizes the end-to-end BER. The choice of a specified path is very important for this strategy, as both factors can be directly related.

For the choice of the appropriate route, the MAODV gives preference to the path where the longest link distance between source and destination nodes is the shortest one possible. Bear in mind that link distance is a different metric than the hop count; it actually represents the physical distance (meters) between nodes. The next few lines should help us study and understand how the distance metric is co-related to the end-to-end BER.

We will assume from the beginning that on every receiving node of each hop there is packet regeneration and re-transmission, and that errors that were created on one hop are not recovered in the next one. So we denote as $BER_{link}^{(i)}$ the BER at the i-th link of the route, which depends on the link's signal-to-noise ratio (SNR) and medium characteristics. The end-to-end BER can, as stated, be written as follows:

$$BER_{route} = 1 - \prod_{i=1}^{n-1}(1 - BER_{link}^{(i)})$$

$$= \sum_{i=1}^{n} BER_{link}^{(i)} - \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} BER_{link}^{(i)} BER_{link}^{(j)}$$

$$+\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\sum_{z=1,z\neq i,z\neq j}^{n} BER_{link}^{(i)} BER_{link}^{(j)} BER_{link}^{(z)}$$

$$-\ldots$$

For the BER to have a reasonable value at the end of a several hop route, the BER of the direct link has to be very low, ideally $BER_{link}^{(i)} << 1, \forall i$.

As the sum of all link BER is the biggest contributing term for the route BER, we can approximate it, very accurately, to:

$$BER_{route} \backsimeq \sum_{i=1}^{n} BER_{link}^{(i)}$$

In case we consider having a *Binary Phase Shift Keying* (BPSK) over and *Additive White Gaussian Noise* (AWGN) channel, we can write the BER of the link as:

$$BER_{link}^{(i)} = Q\left(\sqrt{2SNR_{link}^{(i)}}\right) \ , \ Q(x) \triangleq \frac{1}{\sqrt{2\pi}} \int_{x}^{+\infty} e^{-\frac{t^2}{2}} dt$$

where $SNR_{link}^{(i)}$ is the Signal to Noise Ratio at the end of the i-th link. So, if we keep in consideration the BPSK, the link SNR can be re-written as:

$$SNR_{link}^{(i)} = \frac{P_r^{(i)}}{P_{thermal} + P_{int}^{(i)}}$$

where $P_r^{(i)}$, $P_{thermal}$ and $P_{int}^{(i)}$ are the received, the thermal and the interference power values at the end of the i-th link.

We have to stress at this point, that the thermal noise is the same for all the nodes. If we assume two-ray ground path loss [20] , the following expression if considered correct:

$$P_r^{(i)} = \frac{\alpha P_t}{d_i^4}$$

P_t being the transmission power, α a propagation constant that depends on the receiver and transmitter antenna gains, antenna heights and the carrier frequency, and d_i the length of the i-th link.

Summing all of this together, we end up with an approximation of the general BER of the route:

$$BER_{route} \backsimeq \sum_{i=1}^{n} Q\left(\frac{1}{d_i^2}\sqrt{2\frac{\alpha P_t}{P_{thermal}+P_{int}^{(i)}}}\right)$$

The interference power $P_{int}^{(i)}$ depends on how the nodes are distributed and also on the MAC protocol. If we assume that the network topology is homogeneous, $P_{int}^{(i)}$ will be very similar for all the links of a certain route. Thus, as $Q(x)$ is a rapidly decreasing function of x, if one hop had a length considerably bigger than the others, in a route, the BER for that hop would be the highest possible and, finally, the resulting BER of the route will be

$$BER_{route} \backsimeq Q\left(\frac{1}{d_{i_{max}}^2}\sqrt{2\frac{\alpha P_t}{P_{thermal}+P_{int}^{(i_{max})}}}\right)$$

where $i_{max} \triangleq argmax_i\{d_i\}$, so the i_{max}-th hop is the longest one.

To conclude, the BER has to be kept small in order to have successful communication. After a given amount of BER, code correction can't handle the repairing of the packet and the frame is lost.

We can correctly say that the routing strategy that minimizes the BER of the route would be, in fact, the one that will prioritize the choice of the route that contains the lowest possible maximum hop length or, simply put, the shortest possible distance between two given nodes.

Figure 4 shows the actual difference between route choosing strategies. In our particular case, MAODV and AODV.

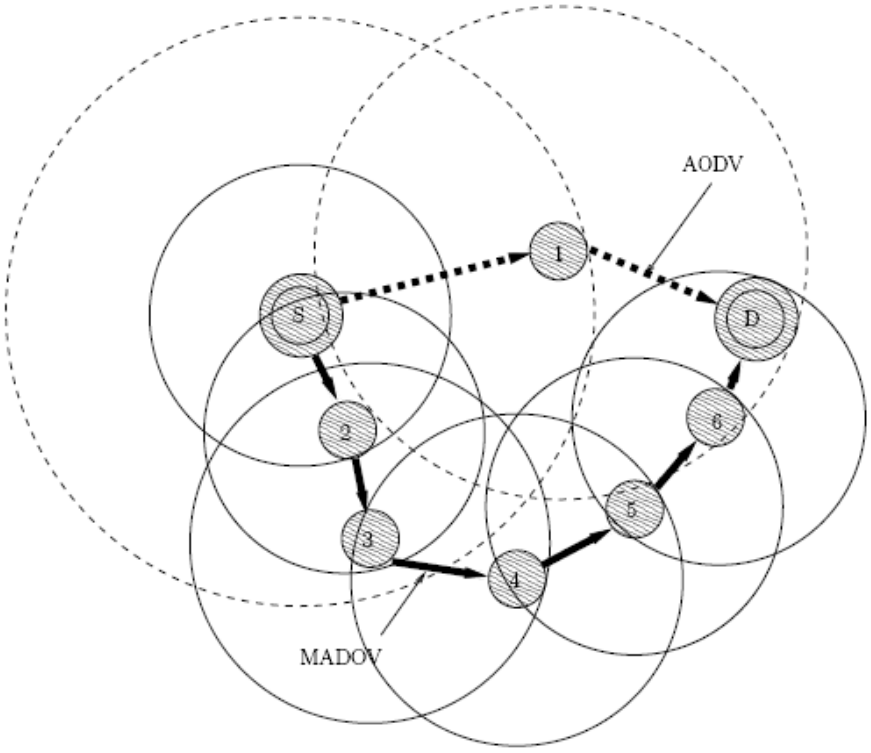

Figure 4: Routing strategies: MAODV vs AODV

5.3 Distance and transmission power

As explained in the previous section, for it to be possible to chose the route that minimizes the BER of a given link, we have to be able to calculate the physical distance between nodes. For this we tried to understand the relation between the signal power at the receiver and the distance separating both nodes.

We know that the distance can be translated into a function of the reception power in the same proportion as shown in the next graphics.

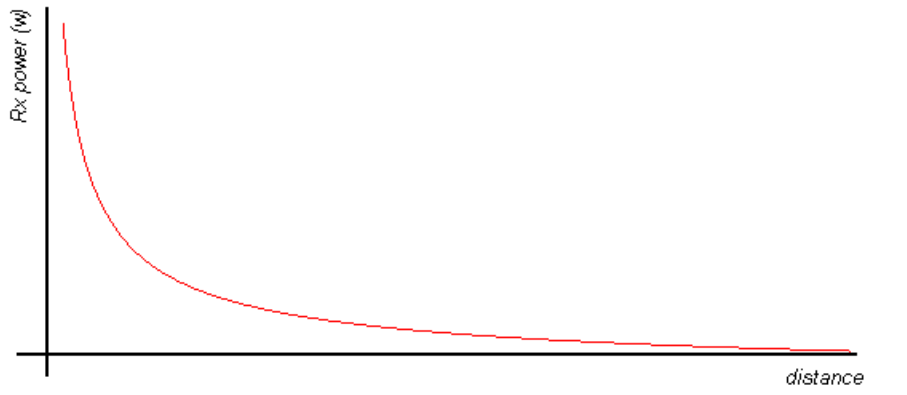

Figure 5: Relation between distance and received power (watts)

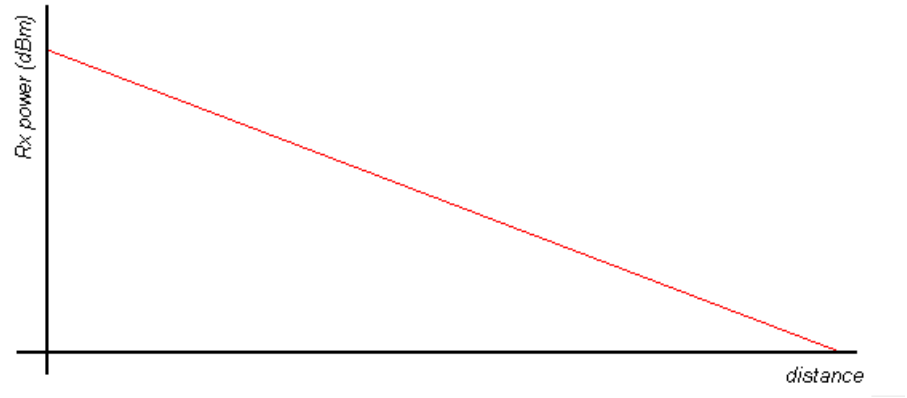

Figure 6: Relation between distance and received power (dBm)

To discover the physical distance we have to also relate the transmission power value to the reception power value. Like that we have

$$P_{rx}[dBm] = P_{tx}[dBm] - 10\ .\ x\ .\ log(\frac{d}{d0}) + N(0, \sigma)$$

where $P_{tx}[dBm]$ is the ***Friis Transmission Equation*** for the power, x is the ***path loss exponent*** which depends on the transmission anbience (normally $x \backsimeq 3.5$ for indoor anbiences), $d0$ being base distance, which should be approximately 1 meter ($d0 \backsimeq 1m$) and finally, $\sigma \backsimeq 3dB$ for these cases.

From the Friis model, we know that

$$P_{rx} = P_{tx} \left(\frac{G_t\ .\ G_r\ .\ L}{(4\pi\lambda^2)^2} \right)$$

where G_t is the gain obtained from the transmission antenna, G_r th gain from the reception antenna, L the loss in the transmission and λ the wave length of the specified signal.

Thus, assuming that the transmission power has a standard value (in our case we set it at 17dBm as we will see ahead and explain the reasons), and that we are looking for an average equation, making $N(0, \sigma) \backsimeq 0$ also on average (zero-mean Gaussian variable), we can deduct a standard average equation to relate the reception power to the distance

$$\overline{P_{rx}} - 17dBm = -10 \,.\, 3.5 \,.\, log(d)$$

thus

$$d = 10^{\left(-\frac{\overline{P_{rx}} - 17dBm}{35}\right)}$$

5.4 Energetic concerns in wireless networks

During the whole development in the 802.11 technology, parameters like bandwidth, radius and latency have been considered prioritary over others, such as the energy consumption. As the original structure plan for WLANs was represented by an infra-structured mode (AP mode), power outlets or solutions like POE were a given fact, and did not weigh on the vendors solutions.

With the maturement of the whole wireless network concept, new mobile solutions that could not depend on fixed power sources were created. Several embedded systems, outdoor retransmitters or other systems that could not be associated with a fixed power source would depend on batteries or other alternative power sources. It became quite evident that when the power source is either costly or in short supply, energy efficiency is of paramount importance. In some wireless network applications, energy is actually entirely nonrenewable and is thus an overriding constraint for the design and operation of the network.

One of the most challenging examples for energetic concerns are in fact ad hoc networks. As previously referred, these are networks that until recently were of interest

mostly in military applications, but now there is also widespread commercial interest for cases of sensor networks in which multiple micro-embedded devices are interconnected in autonomous systems that are deployed only once. The latter clearly survive only as long as their original batteries are capable of providing the energy they need. But all ad hoc (multi hop) networks are critically dependent on the rate of energy consumption. In fact, it is in this class of networks that all aspects of energy efficiency are most clearly displayed.

In this project the energy concern is actually one of the most important issues. The ability to control the transmission power (chapter 3.4.1) is used in order to obtain two desired effects: a higher performance, through a higher packet delivery rate, and lower power consumption.

5.5 Performance enhancement techniques

In order to achieve higher performance measures in ad-hoc networks, we are going to explore two different techniques. These techniques can be used in several cases and in the case of QOS in any kind of network, wired or wireless. For this project we are going to use one of these procedures, more specifically the Power Control, as it is a lower level implementation technique, and QOS can be applied after the algorithm implementation is done. In fact QOS can be applied in almost any case, without interfering directly with any routing protocol.

5.5.1 Power Control (PC)

As was referred earlier, controlling the radio transmission power can be a valuable tool to boost performance and reduce power consumption. In our specific case we are going to adjust the transmission power according to the types of messages being sent.

The assignment of the power value depends on two parameters. The first one is the kind of message being transmitted. They might be AODV control packets (RREQ, RREP, RERR,...) of normal traffic packets being routed to their destinations. In case it is a control packet, the transmission power will be set to the maximum value, which we established as being 17dBm according to regional standards and hardware limitations so to create compatibility for as many devices as possible. In the second case (regular traffic) the

transmission power will be set depending on the physical distance to the next node on the route. We have defined 8 different power levels depending on the distance to the next destination. These values are attributed according Table 3.

Table 3: Power settings

power value (dBm)	Relative distance
1	0 to 10
5	10 to 20
9	20 to 35
11	35 to 50
13	50 to 60
15	60 to 70
16	70 to 80
17	80 to 92

As known, the relation between the transmission power and the covered distance is a power of 2. It can be roughly translated by $power^2 \rightarrow acheived \;\; distance$.

Using this technique showed a lot of potential in simulations, meaning that if used in combination with our chosen protocol, a good load balance algorithm and a good set of QOS rules we could achieve arround 20% performance increase in general network usage cases.

5.5.2 Quality of Service (QOS)

It's not easy ensuring quality on wireless LANs. Because of a WLAN's limited capacity and shared medium, you cannot over provision it like a wired Ethernet LAN. Furthermore, WLAN traffic patterns can be unpredictable, so enforcing quality of service is tricky, especially when wireless traffic crisscrosses the wired Ethernet infrastructure.

On a land line Ethernet and IP network, you can use QoS prioritization schemes– 802.1d and DiffServ, for example–or throw more bandwidth at the problem to ensure time-sensitive applications like voice and video get sufficient headroom. As these applications– which are susceptible to unpredictable bandwidth, high latency and jitter–move to WLANs,

so too must QoS schemes.

VoIP (voice over IP) will likely be the most common time-sensitive application for a typical enterprise, and the one most in need of QoS. Enterprise adoption of VoIP has not exploded, of course, but wireless VoIP makes it more appealing, especially if a WLAN infrastructure is being deployed anyway for conventional data applications.

Wireless VoIP gives workers who are not regularly at their desks more flexibility, and it can reduce operational costs associated with the use of cell phones and private radio walkie-talkie systems in the enterprise. Warehouse, distribution, retail and health-care operations have used wireless VoIP for a number of years, and it will become increasingly popular over the next several years as both VoIP and WLAN technologies mature and their prices decrease. In fact, the percentage of large enterprises deploying wireless VoIP is predicted to increase from the single digits today to 33 percent in 2006, according to Infonetics Research.

Wireless QoS can increase other applications as well, such as guest access, where you offer your clients, customers or other visitors wireless access over your WLAN while they are on site. QoS here gives your internal users higher priority than visitors, and likewise, some internal users take precedence over others.

Then there is the consumer market, where Wi-Fi has been a huge hit as a home networking technology for sharing printers, files and broadband Internet connections. QoS lets home users prioritize how bandwidth gets split among these kinds of operations. WLAN QoS also is necessary for multimedia home entertainment, where moving digitized multimedia content across home systems is gaining steam.

However, QoS is more complex in an enterprise environment because applications are more varied and the physical scale of WLANs is greater. With QoS standards slow to emerge, some organizations have taken pragmatic approaches to ensuring their applications get the bandwidth they need. Some hospitals, for example, deploy multi band 802.11 a/b/g network infrastructures: They dedicate the 5 GHz 802.11a system to data applications and the 2.4 GHz 802.11b/g system to voice.

For implementing a set of rules or strategies for QOS policies, we suggest that MPLS, DiffServ and IntServ frameworks are used. Along with an appropriate set up of the network, these tools should provide a valuable, though rather limited, set of rules for good traffic shaping.

6 The MAODV-PC Project

For the network to function at its full capacity it is imperative that the routing systems and overall processing power requirements are previously studied. Insufficient processor power and memory requirements would lead to latency increase and bandwidth reduction, and excessive signal transmission power or lack of medium synchronization would translate into increase of signal noise and interference for surrounding networks or nodes.

A good architecture is also fundamental. Bad software planning may translate into higher processor consumption and therefore more time to execute the same operations. Operation priority should be evaluated as well, it is imperative that the objectives are always in mind and to know which are the most important and crucial parts of the architecture.

In this particular case, in a cross layer design, all of the previous advices are crucial. Cross layer design has become of great interest for the development of wireless networks. They represent an opportunity to integrate different layers in order to achieve better results. However, such cross-layer design can run at cross purposes with sound and longer-term architectural principles, and lead to various negative consequences [11].

6.1 Routing Protocols

AODV vs MAODV-PC

As previously mentioned, the choice of the appropriate routing protocol for an Ad-Hoc network is a very important step for the proper functioning of the topology.

As concluded by our previous analysis, standard AODV gives preference to lower latency rates as opposed to MAODV-PC, which prioritizes the delivery ratio (which, because of TCP/IP retransmitions means more usable bandwidth). As several standard AODV protocols have been studied and implemented, we will give preference to MAODV in our demonstration.

For a future development of the proposed software, we suggest that a traffic analysis algorithm is developed in order to apply both routing algorithms, depending on the type of traffic. For example, we would use MAODV-PC for HTML and POP, giving preference to bandwidth, and use standard AODV for low latency traffic like VoIP. In this manner

we can have 2 different routes, depending on the type of traffic and QOS classification, thus balancing the load in the network and increasing the useful network usage rate.

6.2 Real Life Demonstration

As *proof of concept* for the proposed protocol we intend to create an application that will use it for real time, real life routing. The created application uses routing and medium controlling tools already available for current systems (IPTables (netfilters), Wireless Extensions, etc). These tools and extensions should provide full functionality coverage for the desired results like route establishment, transmission power control and signal analysis. The implemented application will be able to manipulate these system tools so as to be able to construct appropriate routing tables which will contain cost metric parameters, BER at the end of multi-hop routes, SNR values, transmission power values, neighbour node information, among others.

With this information the application will be able to implement the MAODV-PC routing protocol within the expected parameters.

6.3 System Requirements and Architecture

The standard network will probably be composed of access points acting as nodes, or of other embedded wireless capable devices (possibly portable laptops with wireless interfaces). For simple system programming, a Linux OS is preferable to a proprietary solution.

The system core would be based on a distributed processing and routing paradigm, using mixed architecture of a Linux Kernel module and a user space daemon. For porting purposes, both between different operating systems or different Kernel versions in the same OS, a stand alone daemon is used for all the wireless handling functions, although a single Kernel Module would represent a higher gain in performance and processing efficiency, becoming the best choice for a fixed/infrastructured architecture.

Although we are aware of this, it was our choice to follow a mix architecture (kernel + daemon). This was made for several reasons of which the most important are:

- Stability. - Keeping functions that might rise problems (signal monitoring and power selection), such as signal detection failure, unstable driver code, in a protected environment, making sure that the fundamental functions and the whole system is safe from critical corruption.

- Future work - implementing some new signal functions or update current wireless code would be much easier if working in kernel space for the purpose.

In the application, the daemon part would be required to handle all radio (physical) manipulation (as power control, signal monitoring, BER calculation, etc) as the kernel module handles all of the routing routines (logistic), like route conservation and updates, route building, packet transmission, etc.

The know how for creating this software was given to us, mainly, from the book [12]

Figure 7 illustrates the general architecture of the application.

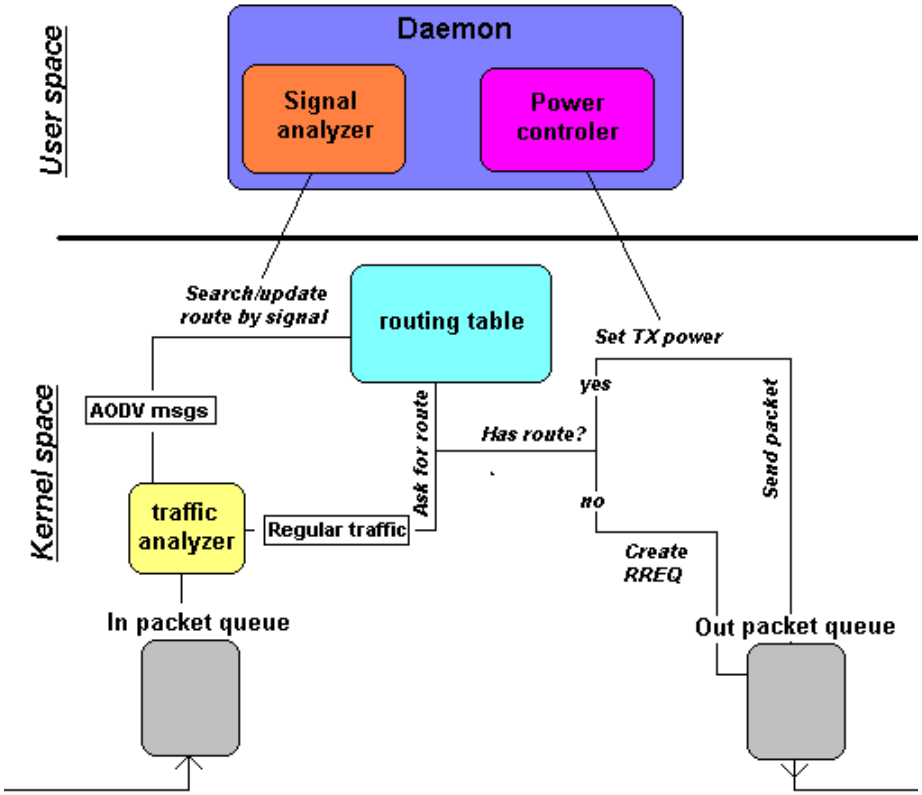

Figure 7: General system architecture

As we can see through the extremely simplified system diagram, both parties, module and daemon, communicate with each other to provide the necessary information for both to execute their function. The Module provides a list of addresses and a power value that the daemon uses. These addresses are constantly monitored by the daemon which sends monitoring information back to the module, like SNR, signal strength and noise level. Besides the signal monitoring, the daemon also sets the radio transmission power according to the value established by the module.

In our system, as AODV and MAODV are reactive protocols, there are two ways of provoking action. One happens when a message or packet is received and the other is triggered by timers, which can be for neighbourhood discovery, route timeout or RREQ waiting timeout...

The whole system can be divided into layers or modules. A good architecture should always be thought as a layer system, in which higher level functions depend on lower level

layers to do their job.

In the next sections we will analyze these layers in detail.

6.3.1 Netfilters and Wireless Extensions

There are also some important parts of the linux base system that are used by the protocol. The netfilters application package and the kernel wireless extensions are used by the application and, as such, should be properly introduced.

Netfilters

Netfilters [24] are used by our implementation to identify many of the events that trigger routing protocol action. A netfilter consists of a number of hooks at various points inside the Linux protocol stack. It allows user-defined kernel modules to register callback functions to these hooks. When a packet traverses a hook, the packet flows through the user-defined callback method inside the kernel module.

There are five hooks defined in the Netfilter architecture, shown as boxes in Figure 8.

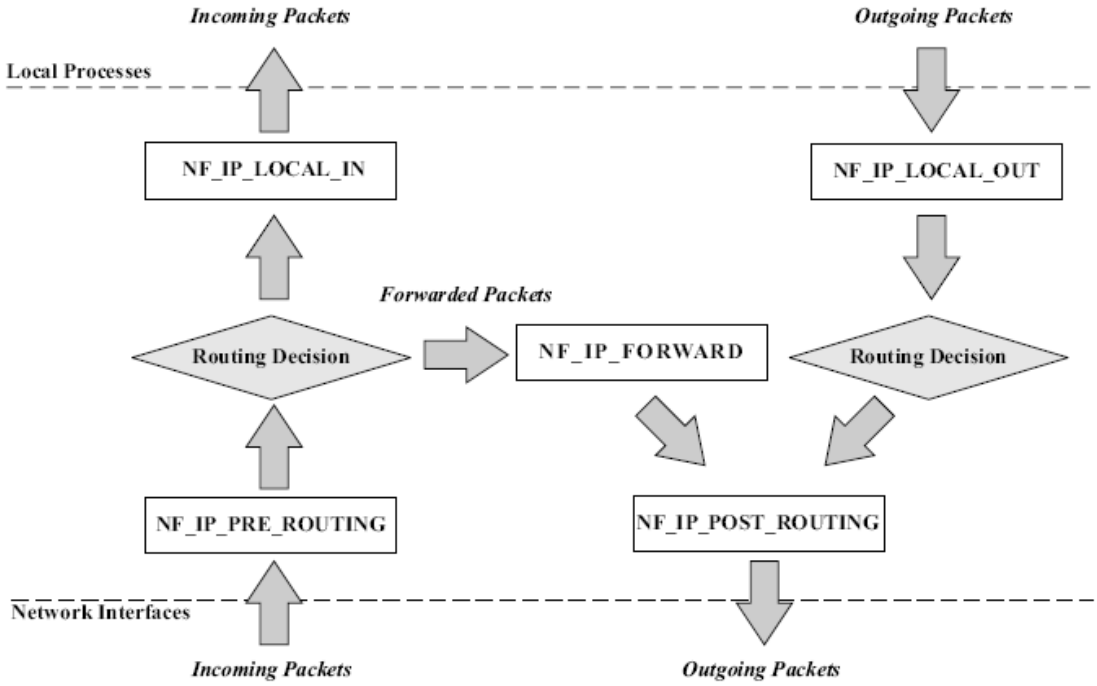

Figure 8: Netfilters hooks

At the top of the figure there are two hooks, NF_IP_LOCAL_IN and NF_IP_LOCAL_OUT. These hooks are for all packets to and from local processes.

At the bottom of the figure there are two hooks, NF_IP_PRE_ROUTING and NF_IP_POST_ROUTING. These are for all packets from and to other hosts on the network. There is also a hook for packets that are forwarded by the current host, NF_IP_FORWARD.

As an example of how packets traverse these hooks, suppose a packet is created by a local process for a remote process. It first traverses the NF_IP_LOCAL_OUT hook. Next, a routing decision is performed to see whether the packet is bound for the local host or another host on the network. In our example, the packet is found to be destined for a remote host, so the packet is passed through the NF_IP_POST_ROUTING hook and then onto a network interface.

In our specific case, the netfilter hooks are responsible for all of the major packet tasks, from capturing the messages, to the system route establishment and outgoing packet queuing.

Wireless Extensions

The Wireless Extensions (WE) [22] are a standard API for wireless drivers and programs. Using the Wireless Extensions, hardware drivers are completely abstracted from user programs and vice versa. Since one of the earliest versions it has been included in the linux source code, being considered a great contribution to the development of the wireless subsystem in the linux operating system.

The Wireless Extensions support come as 3 parts :

- The Wireless Extension (the core API), part of the Linux kernel (mostly defined in wireless.h).
- Driver support, implementing some of those extensions for the specific hardware.
- Wireless Tools (a combination of system tools to manipulate the WE).

The core API is strongly versioned, so that application can detect changes in the API. The version number of Wireless Extensions specify which feature is available through the API, and is completely independent of the version of the tool using it. For example, each version of the Wireless Tools can support a wide range of Wireless Extension versions. We can use "'iwconfig –version"' (existing in the most recent linux distributions) to get all the details on the current system setup.

One common issue is that Wireless Extensions offer only source level backward/forward compatibility, not binary compatibility. Therefore, every time the core API is updated or changed on the system (kernel upgrade), both drivers and tools need to be recompiled. Version 27 and later of Wireless Tools can support multiple versions of Wireless Extensions without the need of a recompile.

Some wireless drivers require a minimum level of Wireless Extensions, and most often a newer version of Wireless Extensions enable additional features in the driver. Usually, the simplest way to upgrade the Wireless Extensions is to upgrade the linux kernel. Most drivers support the newer Wireless Extension only in their most recent version. Therefore, generally to get the benefit of the additional features it is preferable to upgrade the driver.

6.3.2 Packet capturing and analysis

The first layer and the basis for the whole system is packet capturing and analysis. We have to be able to distinguish regular data (IP traffic) from the protocol control messages (RREQ, RRER, RREP...).

For this purpose a system tool is used. The netfilters tool, also known as IPTables, included in the core of the standard linux kernel is responsible for capturing all of the packets sent to the host and inserting them in a queue for the MAODV module to analyze and process. The queue is a type of FIFO stack, in which the packets are processed in the order by which they arrived.

These packets are then removed from the queue by the program, which analyzes the type.

The first case is for regular data packets. The module will identify the destination, and in case the destination is the same host that received the packet, the packet is considered delivered and is returned to the system which handles it and delivers it to the application in cause. In case the destination is a different host, the module will try to find an appropriate route to the destination and deliver the data.

For MAODV control messages, the process is quite different. If a RREQ is received, the system will search for a route and return it to the node from which the message was originated. In case the route does not exist, the RREQ will be re-broadcasted. For a RREP, the system will analyze the route, and in case it does not exist or the metric is better than the available route, it will be added to the internal routing table.

The MAODV message processing belongs to a higher level layer, so it will be discussed in greater detail in the next sections.

6.3.3 Event handling

When the packet is received and analyzed by the lower layer, it is then sent to a handling routine. This routine acts as a task queue, where all system tasks are dealt with. As referred in the previous section, these tasks can be the processing of messages or tasks inserted by the timer triggers. The task queue is actually another FIFO stack where all events are dealt with in the order by which they are inserted. These tasks are then processed by a thread which decides the appropriate action to take. We will go through these tasks or events in the next few paragraphs, starting with the received messages.

Each different type of packet has got its own engine that processes the message and acts accordingly.

The first and most basic case is for regular IP traffic packets. In case an IP packet is received or created by the host, the system will try to acquire a route. If the route does not exist, a RREQ packet is created and broadcasted to the neighbourhood. The host will then wait for a time out (3 seconds in our case, adjustable), so that all replies, or at least most of them, can be received and the best route (the one that travels through the closest neighbour) can be chosen for the relaying of the data. This way we can always assure that the best route is chosen according to the MAODV algorithm, which specifies that the best route is the one through the closest surrounding node.

For the RREQ packets, when received, its engine updates the reverse path to the source, and afterwards searches for an available route to the destination. In case this route is found, it generates a RREP which is sent to the source, but if the route to the destination does not exist, the RREQ is re-broadcasted to the surrounding nodes like in the previous case (IP traffic).

When a RREP is received, the system will search for a previously existing route. In case this route exists, two parameters will be checked; one is the sequence number and the other the distance value. If the sequence number of the received route is smaller than the one belonging to the pre-existent one, then the RREP is dropped and the route is not updated. If the sequence number is greater in the newly acquired route, the distance to the next hosts will be compared, and in case the distance is smaller in the newer route, the protocol will drop the existing path and replace it with the recently received one.

For the timer triggered tasks, they are very specific in purpose.

The first and most complex timer task is the complete reset of all the structures. All of the routes, the neighbouring information and queues are periodically flushed in order to prevent corruption of the structures or that old, no more relevant data is kept indefinitively. This will prevent software corruption caused by memory leaks due to long time functioning. This mechanism will guarantee that the system will be able to work properly for long periods of time with no need of any kind of manual maintenance or human interaction.

The following trigger is the periodical Hello message broadcast timer. Its purpose is to send out hello messages periodically in order renew the neighbour list. In this way we can make sure that the neighbour list is accurate at all times. If the neighbour list is not correct at all times, we will have broken routes, and might be sending packets to nodes

which no longer exist thinking that they will be received. When a hello message is sent, all neighbouring nodes will include the host in their local node list and reply with another hello message so that the original host knows of their existence.

The next task is the neighbour timeout routine. Every time a new neighbour is acknowledged by the system, a time stamp is included in that position in the list. Every few seconds the timer will search for timed out neighbours and delete them from the local list. Like this we make the system maintain a fresh neighbour list. This technique prevents no longer existing neighbours from still being included in the routes, corrupting the whole algorithm.

The purpose of the next timer is to periodically scan the routing table for old routes that have to be flushed. All routes have to be refreshed periodically, making sure that non existing routes are not maintained. If an invalid route still exists in the table, it might be chosen for the routing, though it is not correct anymore. This will make the host re-transmit the data and acquire a new route, making the transmission delay unbearable. With this mechanism we are sure at all times that all of the routes included in the table are correct and available at all times.

The final trigger is one of the most crucial for the system, as it handles all of the RREQ procedures. In its procedures there are two very important tasks. The first one is to resend unreplied RREQ periodically (it will only keep sending the RREQs if *no* RREP have been received), making sure that, even if data is received for relaying and no route exists, as soon as a route is available to the destination it will be discovered and the data will be sent. The second task is to wait for a maximum time for all the RREPs to be received. This timeout is only active if at least one RREP has been received, making sure that it will not drop the RREQ before an available route is acquired.

6.3.4 Routing table

The routing table is, obviously, the core of the algorithm. It is here that all the routing information is stored. It is implemented through a linked list acting as a vector. This solution was chosen instead of other structure types, as a hash table, due to its better performance, making the manipulation of the table quicker than it would be if another kind of structure had been chosen.

The information stored in the internal table includes the destination's IP address and

net mask, the route's current sequence number, previous sequence number (maintained to distinguish intermediate routes), the number of hops to the destination (to preserve the compatibility with regular AODV protocol), the distance metric to the next hop, the IP address of the next hop, the route's lifetime (time until it is dropped from the table and the identification of the device to use for this route (useful for future development, gives the possibility to use several devices in different channels in the algorithm, in order to achieve lower RFI, communication parallelism and therefore higher available bandwidth).

However, the internal routing table is not responsible for the actual routing. It manipulates the netfilter kernel system (IPtables) in order to create a system functional routing table. For every route that it contains it creates a new IPtables rule, which makes the system core (Linux kernel in this case) take care of the actual physical packet routing. When the route is dropped by timeout or is updated and replaced by a better one, the internal table checks the existing IPtables rule for that specific case, removes it from the system and creates a new one to replace it, if needed.

The internal routing table is not only responsible for storing the current routes, but also includes dynamic functions that create new routes, updates the old ones and deletes the paths that are no longer in use, as well as minor maintenance routines. It also includes a somehow complex locking mechanism to prevent table corruption caused by simultaneous writes and reads from and to the table structure.

6.3.5 Procfs bridge

As previously mentioned, one of the parts of the system is a signal daemon which is responsible for monitoring the signals of all of the surrounding nodes and setting the transmission power value. To provide a communication channel between the kernel module and the daemon, a procfs ***(proc file system)*** entry is created (file in the
proc directory in the linux base system). This entry allows the module to send a list of MAC addresses and an instantaneous power value to the daemon as well as provides the means for the daemon to send the signal statistics back to the module.

This kind of entry is the standard and correct way to transport data between the Linux kernel space and the user space. The kernel space is composed of all of the memory regions belonging to the linux kernel, where only the kernel itself can operate. By contrast, the user space is exactly the opposite. It is the memory space reserved only for user side

processes. The data transition between both spaces is a complex process, due to absolute restrictions in the kernel memory and due to the way memory pointers are handled in both spaces. The kernel uses only physical memory, never using swapped hard drive space as memory replacement. In fact, the kernel does not even have this concept, as it is only reserved for user space. So, if we used direct memory mapping between Kernel space and User space we could have serious memory corruption problems and access restricted or unavailable memory regions.

We will now describe how the data transmission occurs.

From the kernel to the daemon, the data is sent in ASCII (text only) so that a user can, at any time, see what physical addresses are being monitored and what is the transmission power value. The addresses and the power value are sent in separate lines, being the power value in the first line and the addresses in the following ones as shown in the example in Table 4.

Table 4: Proc - Kernel to Daemon

attribute	value
power level	9
address 1	00:13:f7:11:b1:68
address 2	00:30:ab:09:5a:89
address 3	00:13:f7:16:cc:58
address 4	00:30:ab:09:5a:c8
...	...

In the opposite sense, from the daemon to the kernel, the signal statistics travels in custom data structures, declared specifically for the effect. For each physical address, the daemon will have to send 5 parameters, the MAC address to which the statistics belong, the signal level, the noise level, the calculated quality level (close to SNR) and a flag that indicates the validity of the signal values, making it very unperformant to use ASCII data for the transmission. Binary structures are used so that there is no need for binary to text conversion on the daemon side and text to binary on the module side.

For this transmission two structures are used. The main one contains an integer indicating how many hosts are being monitored at the time and an internal structure array which contains the actual statistics, one per monitored host. The structures are organized

as shown in Figure 9 and Figure 10.

```
/*
 * Structure used for delivering collection of statistics
 */
typedef struct all_stats
{
   int          number;              //Number of statistics collected
   signal_stats stats[IW_MAX_SPY];   //Individual statistics array
} all_stats;
```

Figure 9: The main structure

```
/*
 * Structure used for delivering single signal statistics
 */
typedef struct signal_stats
{
  char                      mac_addr[MAC_LEN];  /* Original hosts MAC address */
  unsigned short int        quality;            /* Signal quality (Sig/92) */
  unsigned short int        noise_level;        /* Noise level (negative) */
  unsigned short int        signal_level;       /* Signal level (negative) */
  int                       result_state;       /* negative if error, 0 for success
} signal_stats;
```

Figure 10: The individual statistics structure

6.3.6 Signal Daemon

The user space daemon was developed with two distinct objectives in mind. The first is to collect wireless signal statistics, like signal level and noise level, from the surrounding nodes, thus giving us the effective SNR; the second objective is to set the radio transmission power for a specific value according to the calculated physical distance to the receiving node.

The daemon collects this information so that the module can use it for the choice of the appropriate route through the closest neighbour.

Although it has a very important function, its algorithm is very simple. It works in a loop, and on each iteration it checks the
proc entry, reading the power value and the address list for the hosts to monitor, sets the transmission power and adds the new addresses to the wireless extension spying functions. After it is done reading and setting up the new information from the module, it asks the kernel and the wireless extensions for the signal statistics for the selected hosts and sends back this information to the module.

Manipulating the Wireless Extensions - The WUAPI

For the daemon to be able to set the Wireless Extensions, it has to be able to communicate with the kernel. This communication is made through a series of system IOCTLs which allow an application to send and receive information to and from the kernel through some structures specially designed for the purpose.

To manipulate these IOCTLS, a series of functions have been declared in the kernel system file *wireless.h*. Through these functions we can check if the hardware supports a specific extension, set that same extension, check the result and collect the information.

The available IOCTLs are specified in Appendix A.

The main IOCTLS used in our system are those belonging to the SPY extensions and to the power stings, as expected. They are the following:

- SIOCGIWTXPOW - Tx power value GET
- SIOCSIWTXPOW - Tx power value SET
- SIOCGIWSPY - Spy extension GET
- SIOCSIWSPY - Spy extension SET

And the two main functions used for the manipulations are the following (other secondary ones are used, but only for stability and checking purposes):

- iw_set_ext(skfd, DEV, SIOCSIWSPY, &wrq) - Setting a specific extension, in this case *SIOCSIWSPY* the the device *DEV*, using the kernel socket *skfd* and with the information contained int the *wrq* structure.

- iw_get_ext(skfd, DEV, SIOCGIWSPY, &wrq) - Getting a specific extension, in this case *SIOCGIWSPY* the the device *DEV*, using the kernel socket *skfd* and with the information contained int the *wrq* structure.

In order to make these operations more transparent for the system, a layered solution as created. To make it easier and more versatile for the program to command and manipulate the wireless extensions, we created a Wireless Extensions User Space API which we called ***WUAPI***. This was thought and coded so that the whole architecture could use direct, generic functions to set the necessary parameters, like *set_tx_power(int skfd, int power)* or *get_spy_stats(int skfd, char addr[])*. Although the main structure for the API has been created, only the necessary functions for the MAODV system were implemented and, in a secondary stage, prototypes for other WE will be added.

The WUAPI was created with the knowledge of Jean Tourrilhes (the creator of the WE and WT) and, hopefully, will be included in the WE homepage after concluded.

6.4 Problems, incidents and solutions

During the development of the application, several problems appeared, as expected in any "big" project. The most important of them will be explained as well as the solutions chosen to solve them.

1 - MAC address request not working

The first problem that arose during the development was that some of the drivers for the used hardware would not insert the correct MAC address of the card in the ethernet header of the packet. Figure 11 shows the standard ethernet frame header.

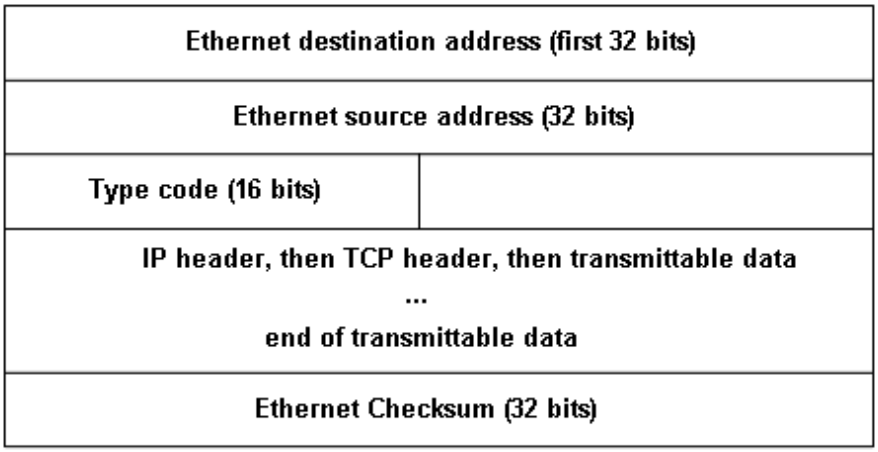

Figure 11: The ethernet frame header

As we know, wireless communications use the same kind of frames as the standard ethernet protocol, which includes the physical address of the source network device. Some of the devices would include an inaccurate address or even none at all, and it was thought that it was a driver layer problem.

For the signal monitoring, the source host hardware physical address is required, so we had to find a solution to discover the MAC address of the neighbouring nodes.

To solve this problem, we included a new field in the MAODV *hello* packet to be able to actually transmit the own address. This way, when the module is loaded, the application requests the device's physical address to the linux kernel and includes it in

all hello messages that are sent to the neighbours. Once the packet is received, the node includes the address in the neighbour's information and uses it for the signal monitoring.

2 - Instability due to signal monitoring on kernel

In the early stages of the application development, the signal monitoring functions where planned to be integrated as part of the kernel module. The signal system was developed using wireless extensions manipulating functions that belonged exclusively to the kernel space.

While testing these routines, we noticed that very severe instability problems arose in some of the test hosts, but not in all of them. The host would cause kernel panic due to illegal memory accessing in random system processes (i.e. kernel swapper process). As the problem would only show up in some of the host, in which the only difference was the wireless hardware, it was apparent that the problem was created by the device drivers once more.

For the purpose of solving the problem, the whole daemon idea was created. We started with several pre-existing examples, working in user space, in which the same problem would not show itself. After the creation of the daemon, all was tested and the stability problems were no longer detected.

3 - Generalization of WE (WUAPI)

While coding the daemon, it was suggested that we should try to generalize, as much as possible, the wireless extensions manipulation functions.

For this purpose we created a special API to allow the use of a generic interface containing all of the wireless oriented functions, like transmission power setting and signal monitoring functions.

The necessity arose from the inability to choose the proper space to place these functions in, and the prediction that, for future development, new wireless functions might have to be created.

This API will be further developed in the future, as at this point it is a parallel, but secondary, project to the MAODV system. It will allow wireless application developers to code new software without having to deepen their knowledge in the lower levels of linux kernel programing.

4 - Data between Kernel Space and User Space

When the Daemon solution was created, it became obvious that we had to find a away to transport the necessary data between user space and kernel space. In fact there are several ways of doing this, but the most common, and still considered one of the most correct, is through a *profs* entry. This consists of a file descriptor placed in the */proc* directory in the linux root tree. This file is completely empty and only points to a shared memory region that is manipulated for this purpose.

When read from or written upon, this file triggers one of two functions in the system process that has created the entry, in our case the MAODV kernel module.

If the access is a read, the triggered function in the module prints the information that is to be received in the user space. This way the process accessing the information can use it at will (*/bin/cat* process for reading the file, for example).

In case a process tries to write in the file, a different function is triggered. It reads the information that has been written, being able to manipulate it as needed, constructing the necessary structures.

7 System testing

To prove that the system works, a simple test plan was developed. It consists of a very simple four node in line topology, where we test both algorithms, regular AODV and MAODV. The test was devised to see if the algorithm worked properly and if it represents any real advantage over the pre-existent systems.

The protocols are going to be tested in the same situations, same hardware and same topology, only differing in the volume of traffic, interference created by surrounding communications and route selection strategy.

7.1 Testing scenarios

7.1.1 First test scenario

Picture 12 shows the first test scenario.

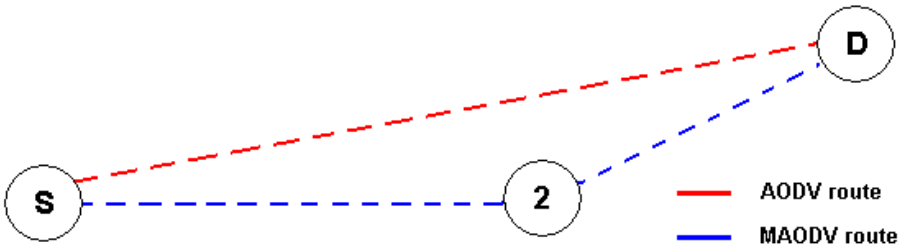

Figure 12: Test scenario (3 nodes)

We set up three hosts at different distances and created a simple FTP transfer. We chose FTP because it is considerably fast for transferring large amounts of data, making it a good benchmark for BER comparison. As it was already referred, having a lower BER means that we have a lower rate of packet loss, and in TCP/IP communications all lost packets are re-transmitted as defined by the protocol. If we study the implications of re-transmission in this scenario, we know that the lower the packet loss, the higher the number of packets successfully delivered at the first try and, therefore, the higher the

effective available bandwidth. We decided to send a complete, compressed linux kernel (± 38Mb), timing the transference and calculating the average link speed. Both protocols were tested in the same situation and the results registered.

7.1.2 Second test scenario

Figure 13 illustrates the second test scenario.

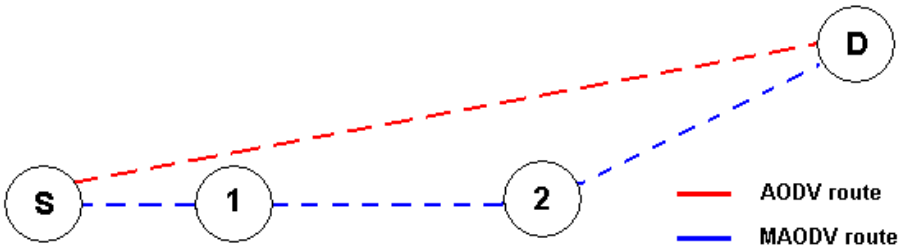

Figure 13: Test scenario (4 nodes)

The second test scenario is in all ways similar to the first case, but instead of having three nodes we use four hosts for our tests. All results we registered and will be discussed in the next section.

7.2 Results and remarks

The performance testing results were registered, and will be shown in this section.

Two parameters were compared, the Round Trip Time (RTT) and the Throughput (Bandwidth).

For the RTT mesuring, ICMP messages were sent, and the reply time was mesured. The distance to cover, was the whole length of the network times 2. ($3hops \times 2 = 6hops$).

For the bandwidth mesurement, a block of 38694KB of compressed data was sent 10 times, and the times we recorded in order to estimate the average transmission speed.

For the regular AODV protocol, there was obviously no diference between the scenarios, as the tests were performed directly from the first node to the last, having no effect the addition of the intermediate node. The results were the following:

AODV	3 and 4 nodes
RTT	3.5ms
Average BandWidth	418 KB/s

In the first scenario, regarding the 3 node scenario, the results were the following for the MAODV protocol:

MAODV	3 nodes
RTT	4.3ms
Average BandWidth	198 KB/s

As for the second scenario, we collected the following results:

MAODV	4 nodes
RTT	5.1ms
Average BandWidth	97 KB/s

As explained in the the first chapter, we cannot compare these results directly as the bandwidth degrades by half at each intermediate node. While the test for the AODV

protocol only has 1 hop with no intermediate nodes, the direct mesured bandwidth was approximately 420 KB/s. As for the the MAODV case, we have intemediate nodes, so in order to compare both results, we will have to estimate the expected bandwidth for a AODV route with the same number of hops.

$$\frac{420KB/s}{2} = 210KB/s(firstintermediatenode)$$

$$\frac{210KB/s}{2} = 105KB/s(secondintermediatenode)$$

If we consider a 10% discount of general routing overhead per intermediate node, we would end up with 189 KB/s of bandwidth in the first node and 95 KB/s on the second, which actualy are values that are very close, if not slightly inferior, to the ones obtained by the MAODV algorithm.

Additionaly, if we think that MAODV algorithm is intended for situations of high network traffic levels and routes with 10 or more hops, in order to reduce the BER, we can predict that, in the same situations and with high number of hops for both protocols, the chosen algorithm will obtain better values than does that could be acheived by the regular AODV.

As another point, the mesured Packet Delivery Ratio (PDR) was superior to 90% in a 4 node route, which can be considered a very positive value and probably higher than the one acheived in a similar AODV route, degrading in a much smaller fraction the regular AODV as the route grows.

At last, the average RTT for the MAODV protocol was even lower than expected (first analysis suggested over 2 times more than regular AODV due to existance of 2 intermediate routing points where packets are processed and re-transmitted), meaning that the routing overhead is considerably small.

As a personal observation, we would like to say that it was a bit of hard work, as we never developed anything like it. In fact, not even close to a level this low. The results were quite satisfying, though, as we realize of much was done and how much we have learned in this period.

The testing also showed that the algorithm reacted as expected. As concerns the power control, it was noticed that the software quickly detected the signal from the neighbours and defined the appropriate power value for the transmissions at any given moment. This value would rapidly stabilize and converge to a fixed value that wouldn't change unless

the ambiance was altered, like nodes moving closer or further away from the source. As for the route update, the algorithm does include new closer nodes in the routes. This was expected, but sometimes it revealed problems, as the signal quality value was not detected accurately at all times.

It was noticed that the software could use some code improving (tweaking) in order to make it a bit more performant. Some of the code can be re-organized or even some functions re-written in order to achieve better performance and to become lighter on the overall system requirements.

In spite of admitting that some of the code could be revised, we still think that the overall performance is good, and that, on the test machines, it was not limited due to processing power or memory consumption. If we would have to appoint any fault for any performance gap, we would probably relate it to some of the wireless equipment that was old and can't keep up with the latest hardware.

The analysis of the code showed that the compiled software is rather compact (28Kb for the daemon and 50Kb for the module) and could easily fit in a firmware of a embedded system to create a commercial product, even one with limited memory and processing power(as the ones shown in the next chapters. For this, it would be quite easy to recompile the code to fit the desired CPU architecture. GCC [23] Cross compilers or tool chains are available for the majority of the architectures, and could be used for this purpose. Generally the architecture for these embedded systems is MIPS which is well documented and easy to install a MIPS compiled Linux firmware.

8 Commercial application

One thing that is rather important to refer before continuing is that, unfortunately, it is very difficult to develop a Microsoft Windows layer for the routing protocol. It depends on modifying the routing subsystem of the OS and probably developing customized drivers for each version of hardware. So, the whole concept relies on these Access Points/Wireless Routers to spread the signal, making it available for regular wireless users. The routing itself would be set up only between the routers that would do it transparently to the end user. For the user, these would act as simple APs re-routing the packets through the network, but would in fact behave as an ad-hoc node to all other routers.

As an example of how this protocol can be used as a real life marketing exercise, and how it could actually be released as a commercial product, this chapter has been developed to show how a full system can be implemented over the proposed architecture.

As we will see in the next chapter, there are already some systems that can be purchased with the intent of building a ado-hoc/mesh topology. Some of these systems are budget oriented, relieving the costs for the end user, or designed to achieve the highest performances possible, cost independent. In our case we will try to find a solution that can cope with both situations. Having a low cost, low performance system for small networks and a medium cost product with considerably higher performance.

The most known low budget systems are based in re-transmission of signals and data, making it very poor in performance. The signals are re-transmitted over the whole network, independent of the destination's location. This means that the network usage costs are immense when compared with systems endowed with appropriate routing protocols that only send the data to the correct destination. In overall cases, specially for small networks where the repeaters are mostly distributed in a straight line, in order to minimize the signal re-transmission over a wide area.

In the opposite cases, the higher performance products normally possess proprietary operating systems and routing protocols, complex and expensive hardware that make them extremely expensive. These solutions normally cost several hundreds of euros, making them inappropriate for small networks or situations with cost limitations.

In our case, we will try to study two separate viable products that would compete with low budget and high performance systems. For both cases the system would rely on the proposed routing protocol, making the cost oriented product much more efficient and performant than the competition's proposals, and the higher performance product much

cheaper than the viable alternatives.

In any of the cases, the operating system is a completely open source (linux probably) [25], as well as the routing protocol and all other tools that would be used, making the software costs almost insignificant.

8.1 Low cost

As mentioned, the software costs are very low, so we will mainly concentrate on the hardware for the systems.

In the cheapest scenario, the studied hardware is composed by very standard and cheap hardware. The chosen hardware is a mips system, making it very easy to alter for the desired purposes.

There are several systems available in the market in our days, but after some research we were able to select the two most adjusted solutions. These are the most know brands, but if a custom solution were required, wireless oriented embedded system boards are preferred. They would require appropriate casing design, but the outcome would be much more original. (ie: www.soekris.com or www.pcengines.ch)

Cisco / Linksys WRT54GL [33]

Product specifications:

CPU	Broadcom 5352 Mips processor operating at 200MHz frequency
Flash memory	4MB Flash memory
RAM memory	16MB RAM memory
Ethernet interfaces	1
Wireless interface	1 × Broadcom wireless NIC (integrated on cpu)
USB ports	None
JTAG port	yes
Serial port	None

Asus wl500gx [34]

Product specifications:

CPU	Broadcom 4704 Mips processor operating at 266MHz frequency
Flash memory	8MB Flash memory
RAM memory	32MB RAM memory
Ethernet interfaces	4 (BCM5325 switch chip)
Wireless interface	1 × Broadcom wireless NIC (mini-pci)
USB ports	2 × v2.0
JTAG port	None
Serial port	yes

Both of the systems are linux based, and can easily be installed with a 3rd party open source firmwares in which the MAODV module and daemon can be installed. Both cost between 75 and 125 per router, or less depending on the dealer, making it as cheap as the lowest priced competition products, but substantially more performant. As a personal choice, the Asus wl500gx would be preferred for the ethernet switching capabilities, higher CPU speed, higher flash and RAM memory, making it ideal for office ethernet switching and much quicker and reliable for the routing due to the cpu speed and high memory.

The open source firmwares can easily be customized and would run perfectly on these systems.

8.2 High performance

The following system is designed to be a high performance routing platform. It should support multi-channeling in all wireless communications, permitting up two 8 times the performance of conventional systems. The hardware is designed to support heavy duty routing and long term functioning without human interference.

In fact, it is considered to be one of the ideal platforms for 4G cellular phone backbone in an IP structure.

Hardware

The MeshCube [35] is a modular hardware design, which can be used for many purposes. It has a Mini-Pci adapter for two MiniPci cards - usually these are WlanCards, but any MiniPci card which is supported by linux can be put in. The adapters can be stacked up on each other, so the system can hold up to 8 MiniPci cards altogether. The cube also has a USB connector which can be used to attach USB devices like WebCams, usb sticks or hard discs.

Operating system

The operating system on the cube is nylon: a small but full featured GNU/Linux system which resembles the Debian distribution in most of the configuration files and utilities. The main difference to debian is that many of the common unix tools (like ls, cd, grep, vi and even a small http server) are implemented in one single binary: BusyBox. This way a lot of space can be saved for other applications. Also the documentation which usually comes with each package has been stripped in order to save space. Other than this it is and feels like a "real" linux box.

The Linux Kernel

The kernel is a standard linux 2.4 kernel [24] (which would be perfect for our MAODV module and daemon), with some additions and patches so it will run on the cube hardware. Most drivers for additional hardware are available as kernel modules.

Management

Configuration and management of the cube and currently only be done by logging in to the cube with ssh. One can login from the wireless or wired interface, by using the IP address of the cube.

Some older versions of the cube come with a web configuration interface. It is NOT recommend to use it, since it has quite a few bugs. 4g systems are currently working on a new configuration system which will provide a consistent interface for configuring the cube.

4G AccessCube [35]

Product specifications:

CPU	AMD Au1500 MIPS processor operating at 400MHz frequency
Flash memory	32MB Flash memory
RAM memory	64MB RAM memory
Ethernet interfaces	1 × 100Mbps
Wireless interface	(see mini-pci)
USB ports	1 × USB 1.1 Host + 1 × USB 1.1 Device
JTAG port	yes
Serial port	yes
Mini-PCI	up to 8 Mini-PCI card expansions for Wireless cards

This system is powerful enough to provide excellent security and encryption, and flexible enough for custom applications and modifications.

Its main features are networking in general, WirelessLan, routing, MeshRouting, autoconfiguration, an emphasis on security (IpSec, VPN) and a compact design to fit on the 32MB flash. It is completely licensed under an OpenSource licence and is based on OpenEmbedded, and the perfect host for a high performance system based on MAODV.

8.3 Firmware development

As said, both solutions are based on linux operating systems. These are always registered as open source under the GNU/GPL license, giving us the liberty to customize them as needed. We could never sell the resultant firmwares because of the legal implications, but we can develop a product (hardware + software), request a patent and sell it as a whole package. This still has some legal implication. For example, it would be required that the original firmware, binaries and specially the source code, are available for download to the open public (ie: product web page, installation cd, ...).

If this set of rules is followed, we can develop two distinct products that should be appropriately viable. Obviously, some marketing studies should be developed to predict the success in the current markets, but considering the high performance and low cost of a well developed system, it is expected that these solutions could achieve a good place in the growing market of mesh/wireless communications.

A completely autonomous system should be very easy to set up. The routing protocol is indeed the core of the system and it is completely developed, just requiring some performance tuning and widespread testing.

So, if we analyze the following requirements, we should be able to develop a small list of services to provide independence to the system.

- DHCP server to provide automatic IP configuration for all nodes
- Active waiting for detection of failing nodes through status messages (nodes send messages to a central server frequently, if the server detects lack of messages, then the node is considered to be down and maintenance can be provided)
- Main web interface for UI, configuration and error reporting (making maintenance very simple, one person can survey and maintain hundreds of nodes)

With these simple services, the network can be set up in a completely automated manner. It would just be required for a maintainer to place the hardware and turn on the power. The routers would automatically detect the adhoc network, require a IP address from the DHCP server, set up the routing protocol and the status messages, providing the wireless service to the users immediately.

9 Some similar commercial systems

9.1 MiLAN AccessG Access Point

This is a commercial grade, SNMP-managed Wi-Fi Access Point with additional capability to be manually configured into a Wireless Distribution System (WDS) [36] [37]. This means that a tree-structured set of wireless interconnection links can be set up between Access Points to eliminate the need to wire them together with Ethernet cable. As WDS is (always, for all manufacturers) manually configured, there is no ability to provide automatic link recovery or fail-over in the event of an equipment or link failure. On the other hand, because WDS is very simple to setup, it is an inexpensive way to connect Access Points.

Main characteristics:

- MiLAN Access Points provide Wi-Fi client connectivity and can be configured to form a Wireless Distribution System
- Each radio allows Wi-Fi client devices to be connected to the network
- Each radio can be configured to transmit all client data packets to one or more Access Points
- Manual configuration of the Access Point distribution system links must never loop back on itself
- The WDS topology is like a tree with multiple branches springing from a single point
- There is nothing automatic about WDS; all configuration is manual
- The WDS "pipeline" has no intelligence; all traffic is retransmitted through the system without filtering
- There is no redundancy in WDS; if a link fails then all Access Points "downstream" are cut off from the portal to the Distribution System
- Compared to FireTide and Tropos, the MiLAN WDS solution is the least expensive

- MiLAN WDS is ideal for single point-to-point links between two or three locations:
  - You can avoid running Ethernet cable when fault-tolerance is NOT a critical issue (ie: hotels, schools, public networks)
  - Perfect for creating a point-to-point link across the street or between two buildings up to 3 miles apart
  - Easy extension of a Wi-Fi network to encompass a peripheral location without the need to run additional Ethernet cable

9.2 FireTide Mesh Router

This is a commercial grade device that is NOT a Wi-Fi Access Point (although it uses standard 802.11 radios.) The easiest way to understand the FireTide Mesh Router [14] is to consider it as a simple Ethernet switch. The unit has a 3-port Ethernet switch, with 3 Ethernet ports, built in. Normal Ethernet devices are connected to the Mesh Router, and these can include Internet gateways, IP video cameras, and even standard Wi-Fi Access Points. The FireTide Mesh Routers automatically discover each other and create a "cloud" of connectivity between themselves. In this way all of the Ethernet devices attached to the various FireTide Mesh Routers become part of a single Ethernet LAN. While more expensive than a WDS system, FireTide Mesh Routers provide automatic configuration and automatic fault recovery as well as intelligent traffic routing through the interconnected mesh.

Main characteristics:

- FireTide Mesh Routers automatically configure themselves to form a redundant, fault-tolerant wireless Mesh Distribution System
- There is no provision for Wi-Fi client attachment to the Mesh; it is strictly FireTide talking to FireTide and pass Ethernet data from Ethernet-wired devices plugged in to the FireTide Mesh Router

- These radios each have an Ethernet hub built in so Ethernet devices can be connected to them (computers, switches, cameras, and standard Access Points through and Ethernet cable)
- Data arriving on the Ethernet hub ports is intelligently forwarded along the correct Mesh Router hop path
- Mesh Routers automatically determine the best path configuration
- At any given time the topology is a non-looping tree, but the matrix of Mesh Routers can automatically reconfigure and establish new paths in the event of a link failure
- Compared to MiLAN and Tropos, FireTide is at the middle price point
- FireTide Mesh Routers are the ideal solution when a fault-tolerant, redundant matrix of Ethernet connections is required for three or more end-points. Remember that FireTide Mesh Routers create a wireless backhaul "cloud" into which Ethernet devices can be attached; they do NOT allow Wi-Fi client devices to connect to them and they do NOT create a Wi-Fi HotSpot/HotZone by themselves:
  - You can avoid running Ethernet cables when a fault-tolerant, redundant system is critical (ie: emergency response, public safety, mission-critical LAN connectivity)
  - Interconnect building floors up the elevator shaft, stairway, or riser (equipment failure on one floor will not disrupt the floors above and below)
  - Build a fault-tolerant mesh for video surveillance when client Wi-Fi requirements are minimum or nonexistent (ie: parking lots, public parks, shopping malls)
  - Create a building-top mesh when more than four buildings must have Ethernet connectivity with redundant point-to-point or point-to-multipoint systems (corporate campus networks, university campus environment)

9.3 Tropos Metro-Cell Wi-Fi Mesh Router

This is, in essence, a commercial trademark, SNMP-managed Wi-Fi Access Point and a carrier-class wireless Mesh Router combined into a single chassis with a very powerful and very sensitive radio. It is not just the two systems plugged together in a single box. The software controls that are part of the Tropos Metro-Cell Wi-Fi Mesh Router [16] individually manage roaming of users throughout the mesh. Hence, a user receives an IP address in one part of the mesh and, if they roam into another cell in the mesh (with a different subnet) the system creates an IP tunnel and the users maintains the same IP address no matter where they roam. Moreover, the Tropos Metro-Cell Mesh Routers are managed from a central console that includes error logs, alarms, and alarm notifications. While a more expensive per-unit device than either a MiLAN Access Point or a FireTide Mesh Router, the features and capabilities of the Tropos Metro-Cell Wi-Fi Mesh Router create an attractive overall cost of ownership when the products are applied as a solution to the right problem.

Main characteristics:

- Tropos radios are dual function devices: a Wi-Fi Access Point for client connectivity AND a high performance wireless Mesh Router for automatic configuration into a fault-tolerant, redundant wireless Mesh Router distribution system
- Each radio allows Wi-Fi client devices to attach to the network
- Data arriving on the Ethernet hub ports is intelligently forwarded along the correct Mesh Router hop path
- Mesh Routers automatically determine the best path configuration
- At any given time the topology is a non-looping tree, but the matrix of Mesh Routers can automatically reconfigure and establish new paths in the event of a link failure
- The Tropos price point is higher than MiLAN and FireTide, but the overall cost of ownership may be significantly less when Tropos equipment is applied to the right application
- Tropos Metro-Cell Mesh Routing technology is the ideal solution for the creation of large Wi-Fi client HotZones and HotSpots where centralized management of the

entire distribution system along with the entire client Wi-Fi system, coupled with ease of installation, is a key factor.

- Community Wi-Fi
- Outdoor venues (ie: RV parks, marinas, airport ramp areas and passenger terminals)

10 Conclusions

The main goal of this thesis was to study and evaluate the concept of MAODV routing protocol and also to see if adhoc technology can increase the potential of wireless networks in wide area use. This has been done through the implementation of the MAODV algorithm. With this tool we managed to observe that the concept is applicable and could set a standard for cross layer wireless routing protocols in the future.

Generally the evaluation gave us good feedback. Meaning that it can be used to further understand the problems within the wireless paradigm and in its linux implementations, and the measures taken that will improve it in the future.

As a robust and cheap technology, Wi-Fi is likely to become increasingly extensive in public locations and private homes to provide high-speed local connection for portable devices such as lap-tops.

More generally, radio technologies such as wireless lans or mesh/adhoc systems might be the way to provide fast Internet access to rural areas where ADSL or cable modem systems are not economically feasible.

Furthermore, as even more renowned network solution companies are taking interest in wide area metropolitan wireless mesh networks, it becomes apparent that it is a growing market in which research and investment is useful and advised.

For a future development of our work, we realise that the application is still far from a release state and that the whole code should be deeply tested and some parts revised. The structure seems to be well developed and should be an appropriate "framework" for the future developments, whether they come from new implemented functions or from code corrections.

Although we consider this algorithm to be already a big step in modern wireless adhoc routing protocol development, we have thought of a few techniques that could be applied to increase further the advantages. After careful consideration we have selected two specific actions to impressively boost performance and reduce even more the BER.

- ***Usage of several radios per access point/router.*** This would actually guarantee that the bandwidth would not decrease in each hop by having a radio for receiving data from other nodes, a radio for transmitting/relaying data to other nodes and another radio to act as the actual Access Point. This way we would preserve full throughput at all times (up to 54Mbps on each link).

- ***Multi channel parallelization.*** As the major objective is to minimize the BER between nodes, we think that the use of non-overlapping communication channels (slightly different frequencies) in nearby links would completely solve the RFI problem, reducing the BER to irrelevant values.

11 References

References

[1] Gianluigi Ferrari, Simone A. Malvassori, Marco Bragalini1 and Ozan K. Tonguz, *Physical Layer-Constrained Routing in Ad-hoc Wireless Networks: A Modified AODV Protocol with Power Control*, http://www.ctr.kcl.ac.uk/IWWAN2005/papers/69.pdf

[2] Luke Klein-Berndt, Kernel AODV, http://w3.antd.nist.gov/wctg/aodv_kernel/

[3] picture taken from: Jason S. King, *An IEEE 802.11 Wireless LAN Security White Paper*, October 2001 http://www.llnl.gov/asci/discom/ucrl-id-147478.html

[4] ABI Research, *wireless Mesh Nerworking*, Original report - ABI Research (Copywrited and available for buyers), http://www.abiresearch.com/products/market_research/Wireless_Mesh_Networking

[5] Openspectrum.info, , *Metro WiFi boom = mesh equipment bonanza (Quoting ABI Research press release on this report) 15/03/2006*, http://www.volweb.cz/horvitz/os-info/news-mar06-012.html

[6] Strix Systems, http://www.strixsystems.com/corporate/default.asp

[7] P. Gupta and P. R. Kumar, *The capacity of wireless networks*, IEEE Trans. on Information Theory, vol. 46, pp. 388-404, March 2000.

[8] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee and Robert Morris, *Capacity of Ad Hoc Wireless Networks*, M.I.T. Laboratory for Computer Science, http://pdos.csail.mit.edu/papers/grid:mobicom01/paper.pdf

[9] S. R. Das, C. E. Perkins, and E. M. Royer, *Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks*, Proceedings of the IEEE Infocom, pp. 312, March 2000. http://www.cs.ucsb.edu/ ebelding/txt/Perkins_PerfComp.pdf

[10] Charles E. Perkins and Elizabeth M. Royer, *Ad-hoc On-Demand Distance Vector Routing*, http://wisl.ece.cornell.edu/ECE794/Mar5/perkins-aodv.pdf

[11] Vikas Kawadia and P. R. Kumar, *A Cautionary Perspective on Cross-Layer Design*, http://decision.csl.uiuc.edu/ prkumar/ps_files/cross-layer-design.pdf

[12] A. Rubini and J. Corbet, *Linux Device Drivers* - 2nd Edition O'Reilly, Jun. 2001,

[13] BelAir Networks, http://www.belairnetworks.com/

[14] Firetide Mesh Networks, http://www.firetide.com

[15] Mesh Dynamics, http://www.meshdynamics.com

[16] Tropos Networks, http://www.tropos.com

[17] Nortel, http://www.nortel.com

[18] C. Perkins and P. Bhagwat, *Routing over Multi-hop Wireless Network of Mobile Computers*, SIG-COMM '94: Computer Communications Review 24(4):234-244, Oct. 1994.

[19] M. S. Corson and A. Ephremides, *A Distributed Routing Algorithm for Mobile Wireless Networks*, ACM J Wireless Networks, Jan 1995

[20] T. S. Rappaport, *Wireless Communications. Principles & Practice*, 2nd Edition, Prentice-Hall, 2002.

[21] Tomas Krag and Sebastian Bettrich, *Wireless Mesh Networking*, O'Reilly Network, Jan. 2004, http://www.oreillynet.com/pub/a/wireless/2004/01/22/wirelessmesh.html

[22] Jean Tourrilhes, Wireless Extensions and Wireless Tools, http://www.hpl.hp.co.uk/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html

[23] GNU Compiler Collection, http://gcc.gnu.org/

[24] Linux Kernel, http://www.kernel.org

[25] The Linux Home Page Online, http://www.linux.org/

[26] Connect802 Corporation, Wireless Data Solutions, http://www.connect802.com

[27] Cisco Systems, http://www.cisco.com

[28] Broadband Wireless Exchange Magazine, *Designing, Planning and Building Mesh Newtorks*, http://www.bbwexchange.com/meshnetworks/

[29] MIT, The Grid Ad Hoc Networking Project, several publications http://pdos.csail.mit.edu/grid/pubs.html

[30] Roofnet project, http://pdos.csail.mit.edu/roofnet/doku.php

[31] Linux and WireLess Mobile AdHoc Networks - MANETs, http://tuxmobil.org/manet_linux.html

[32] MANET Implementations, AODV implementations, http://www.comnets.uni-bremen.de/ koo/manet-impl.html

[33] Linksys, A Division of Cisco Systems, http://www.linksys.com

[34] Asustek Computer, http://www.asus.com

[35] Mesh Cube, http://www.meshcube.org/meshwiki/

[36] Milan Networks, http://www.milan.com

[37] WDS - Wireless Distribution System, http://en.wikipedia.org/wiki/Wireless_Distribution_System

12 Appendix A - Wireless Extension List

/* ——————— IOCTL LIST ——————— */

/* Wireless Identification */

SIOCSIWCOMMIT 0x8B00 /* Commit pending changes to driver */

SIOCGIWNAME 0x8B01 /* get name == wireless protocol */

/* SIOCGIWNAME is used to verify the presence of Wireless Extensions. Common values : "IEEE 802.11-DS", "IEEE 802.11-FH", "IEEE 802.11b"... Don't put the name of your driver there, it's useless. */

/* Basic operations */

SIOCSIWNWID 0x8B02 /* set network id (pre-802.11) */

SIOCGIWNWID 0x8B03 /* get network id (the cell) */

SIOCSIWFREQ 0x8B04 /* set channel/frequency (Hz) */

SIOCGIWFREQ 0x8B05 /* get channel/frequency (Hz) */

SIOCSIWMODE 0x8B06 /* set operation mode */

SIOCGIWMODE 0x8B07 /* get operation mode */

SIOCSIWSENS 0x8B08 /* set sensitivity (dBm) */

SIOCGIWSENS 0x8B09 /* get sensitivity (dBm) */

/* Informative stuff */

SIOCSIWRANGE 0x8B0A /* Unused */

SIOCGIWRANGE 0x8B0B /* Get range of parameters */

SIOCSIWPRIV 0x8B0C /* Unused */

SIOCGIWPRIV 0x8B0D /* get private ioctl interface info */

SIOCSIWSTATS 0x8B0E /* Unused */

SIOCGIWSTATS 0x8B0F /* Get /proc/net/wireless stats */

/* SIOCGIWSTATS is strictly used between user space and the kernel, and is never passed to the driver (i.e. the driver will never see it). */

/* Spy support (statistics per MAC address - used for Mobile IP support) */

SIOCSIWSPY 0x8B10 /* set spy addresses */

SIOCGIWSPY 0x8B11 /* get spy info (quality of link) */

SIOCSIWTHRSPY 0x8B12 /* set spy threshold (spy event) */

SIOCGIWTHRSPY 0x8B13 /* get spy threshold */

/* Access Point manipulation */

SIOCSIWAP 0x8B14 /* set access point MAC addresses */
SIOCGIWAP 0x8B15 /* get access point MAC addresses */
SIOCGIWAPLIST 0x8B17 /* Deprecated in favor of scanning */
SIOCSIWSCAN 0x8B18 /* trigger scanning (list cells) */
SIOCGIWSCAN 0x8B19 /* get scanning results */

/* 802.11 specific support */
SIOCSIWESSID 0x8B1A /* set ESSID (network name) */
SIOCGIWESSID 0x8B1B /* get ESSID */
SIOCSIWNICKN 0x8B1C /* set node name/nickname */
SIOCGIWNICKN 0x8B1D /* get node name/nickname */

/* As the ESSID and NICKN are strings up to 32 bytes long, it doesn't fit within the 'iwreq' structure, so we need to use the 'data' member to point to a string in user space, like it is done for RANGE... */

/* Other parameters useful in 802.11 and some other devices */
SIOCSIWRATE 0x8B20 /* set default bit rate (bps) */
SIOCGIWRATE 0x8B21 /* get default bit rate (bps) */
SIOCSIWRTS 0x8B22 /* set RTS/CTS threshold (bytes) */
SIOCGIWRTS 0x8B23 /* get RTS/CTS threshold (bytes) */
SIOCSIWFRAG 0x8B24 /* set fragmentation thr (bytes) */
SIOCGIWFRAG 0x8B25 /* get fragmentation thr (bytes) */
SIOCSIWTXPOW 0x8B26 /* set transmit power (dBm) */
SIOCGIWTXPOW 0x8B27 /* get transmit power (dBm) */
SIOCSIWRETRY 0x8B28 /* set retry limits and lifetime */
SIOCGIWRETRY 0x8B29 /* get retry limits and lifetime */

/* WPA : Generic IEEE 802.11 information element (e.g., for WPA/RSN/WMM). This ioctl uses struct iw_point and data buffer that includes IE id and len fields. More than one IE may be included in the request. Setting the generic IE to empty buffer (len=0) removes the generic IE from the driver. Drivers are allowed to generate their own WPA/RSN IEs, but in these cases, drivers are required to report the used IE as a wireless event, e.g., when associating with an AP. */

SIOCSIWGENIE 0x8B30 /* set generic IE */
SIOCGIWGENIE 0x8B31 /* get generic IE */

```
    /* WPA : IEEE 802.11 MLME requests */
SIOCSIWMLME 0x8B16 /* request MLME operation;
    /* WPA : Authentication mode parameters */
SIOCSIWAUTH 0x8B32 /* set authentication mode params */
SIOCGIWAUTH 0x8B33 /* get authentication mode params */
    /* WPA : Extended version of encoding configuration */
SIOCSIWENCODEEXT 0x8B34 /* set encoding token & mode */
SIOCGIWENCODEEXT 0x8B35 /* get encoding token & mode */
    /* WPA2 : PMKSA cache management */
SIOCSIWPMKSA 0x8B36 /* PMKSA cache operation */
```

13 Appendix B - GNU/GPL terms

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original

authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at

no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute cor-

responding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License,

they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose dis-

tribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS