Topology Management in Rooftop Wireless Networks

Matt Brown

Department of Computer Science

Hamilton, New Zealand

October 23, 2004

Abstract

The lack of access to broadband Internet services in rural areas of New Zealand has been identified as a significant problem for rural businesses and households. The existing wired telephone network which is being used for broadband delivery in urban areas is around fifty years old in most rural areas and is not capable of sustaining the speeds required for broadband access. This problem has led to the idea of using wireless technologies to provide broadband access to rural areas.

This report investigates existing wireless network technologies and mesh implementations and describes why they are unsuitable for use in a rural environment. Drawing on the experiences of these existing networks a protocol to create a wireless mesh network for use in rural areas is designed. The mesh network created by this protocol is different from existing mesh networks in a number of ways. Each node in the mesh network has multiple wireless interfaces and utilises directional antenna to reduce interference and increase communication range. Additionally the protocol creates a mesh network composed of many layer 2 wireless Ethernet links rather than a single link.

The successful implementation of this protocol within a framework that enables future research to be easily performed is a significant achievement of this project. The results of evaluating this framework and the mesh protocol in two test bed networks are also presented.

Acknowledgements

I would like to acknowledge the support and encouragement that I have received from the WAND group during the course of this project. In particular I would like to thank my supervisor Murray Pearson for his guidance and advice, particularly in the more complex areas of the project.

A big thank you also to Jamie Curtis and Perry Lorier for providing feedback on the mesh protocol and assistance with debugging problems in the mesh framework. Finally lots of love and thanks to my fiancee, Kat, for encouraging me to put my best effort into this project and for taking the time to proof read a long and "boring" report.

Contents

List of Figures

Chapter 1

Introduction

Access to broadband Internet access in rural areas of New Zealand is a significant problem that prevents rural businesses (including farmers) and households from participating in the increasing use of the Internet. The government has acknowledged this problem and is trying to promote the development of broadband capabilities in rural areas through projects such as PROBE[28].

A report by Amos Akmed Swift[20] has identified that existing wired technologies for broadband delivery to rural areas are not adequate and cannot be upgraded to support rural broadband deployment on a wide scale. Similarly existing wireless technologies that have been proposed for use in rural areas have a very high cost or requirements such as line of sight between the transmitter and receiver that make them infeasible for use in New Zealand's rugged rural countryside.

This report examines one emerging type of wireless technology that may be useful in providing a platform for wide scale rural broadband deployment, a wireless mesh network.

1.1 Wireless Technologies

There are two broad classes of wireless devices being used in rural New Zealand today. The first class operate in radio spectrum that requires a specific license to use. These licenses are very expensive to obtain and consequently the deployment of these technologies is solely in the domain of large telecommunications companies. The protocols used with these devices are proprietary. These technologies generally use an expensive base station to cover a large geographical area as discussed in the next section.

The second class of devices operate in radio spectrum that is subject to a general license issued by the government. The general license allows anybody to transmit on a specified set of frequencies provided that the transmit power does not exceed a certain level. These devices usually implement open protocols such as 802.11[5], are widely available and relatively cheap. Unfortunately given the limited amount of radio spectrum available for these devices to operate in and the lack of co-ordination between users, interference is

often a large problem. These devices are very flexible and can be deployed in either the base station model or the mesh model described in the following section.

1.2 Wireless Topologies

1.2.1 Base Station Model

The traditional topology for a wireless network closely follows the model used for television and radio broadcasts. A base station is located on top of a hill and communicates with clients below it. This approach is not as suited for wireless data networks as it is for television and radio signals because they operate in a much lower portion of the radio spectrum. Lower frequency signals are able to "bend" around obstacles much more than the high frequency signals used by wireless data networks. 802.11b and all other wireless technologies in use in rural New Zealand have a very strict line of sight requirement between the base station and the client device. This requirement causes problems for houses and businesses that cannot directly see the base station due to obstacles such as valleys, hills and trees.

In the case of television and radio the communication is one way. This allows the base station to transmit at a very high power to reach nodes that are a long distance away. Wireless data networks require bidirectional communication, each client device must be able to transmit with sufficient power to be able to reach the base station. The equipment needed to be able to transmit long distances can be large and has high power requirements making it unsuitable for deployment in a household situation. Additionally it is not desirable to have every device in the network transmitting at a very high power as it raises the level of background noise and interference that other devices must process, eventually leading to degraded performance.

For this model to work in a rural environment many base stations would be required and each base station would only serve a small number of users. As base stations are typically expensive this approach is not desirable for wide scale deployment.

1.2.2 Mesh Model

An alternative topology for wireless networks that has been gaining popularity in recent years is that of a mesh network. A mesh network is created by linking many nodes together such that each node has a connection to multiple other nodes. There is no base station or central node that everyone must talk to so often the links between nodes are short.

Figure 1.1 demonstrates an advantage that comes from the abundance of short interconnected links found in a mesh network. The three nodes in the bottom right hand corner cannot see the base station due to hills, however they can see some of the other nodes. In this situation a mesh network can provide connectivity through short node to node links where the base station cannot.

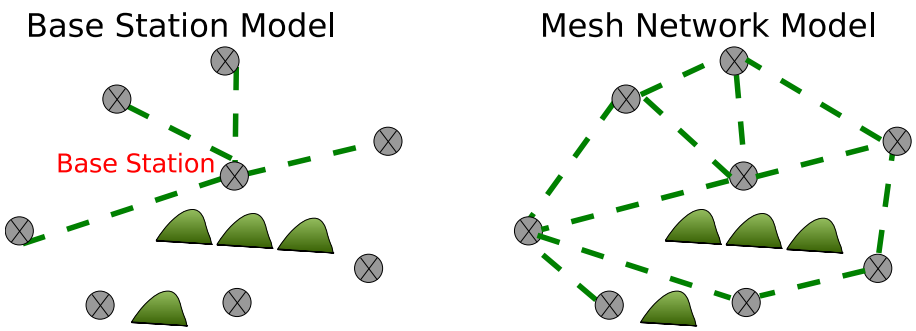

Figure 1.1: Broadcast Topology vs Mesh Topology

The transmit power that a node must use to communicate over a short link to its neighbour can also be much less than it would use to communicate over a longer link to a base station. A mesh network can take advantage of this observation and reduce the transmit power of each node. Reducing the transmit power has the direct effect of reducing interference on neighbouring links and increasing performance.

The mesh network model is heavily used with networks commonly referred to as Mobile Ad-Hoc Networks (MANETS)[12]. MANET research covers many areas related to wireless mesh networks including security [11] and routing performance [4]. Almost all wireless routing protocols available today are a result of MANET research. These routing protocols and their characteristics are discussed more in Section 2.4. Although MANETS are intended for connecting highly mobile devices rather than static locations such as rural households, the existing protocols and strategies may be useful in providing a broadband access to rural areas.

Mesh networks appear to solve many problems associated with traditional topologies and eliminate expensive base stations from the network. However, they are not a perfect solution. The complexities involved in constructing a mesh network are well understood [22]. These include frequent structural change as nodes are turned on and off, node density, interference and limited power availability. Despite these complexities a wireless mesh network would be a significant improvement on the technologies that are currently available for wide scale rural broadband deployment.

1.3 CRCnet

Over the past 3 years the CRCnet project[8] has shown that a successful rural wireless network can be constructed using point to point 802.11b links. While point to point links provide a very high level of performance, such a network is time consuming and expensive to build and requires a significant degree of experience to maintain. It is not a suitable topology for a wide scale deployment to rural houses and businesses.

This project aims to build on the considerable experience that members of the CRCnet project have gained in the design and operation of rural point to point 802.11b networks. This experience will be used to design and implement software that will create a high performance wireless mesh network suitable for wide scale

rural broadband deployment.

1.4 A Wireless Mesh Network

The experience gained from the design and operation of the CRCnet project has identified the following factors as being crucial to the success of a rural mesh network project.

Inexpensive Hardware

The use of inexpensive hardware to construct the mesh nodes means that they can be quickly and easily deployed on a wide scale. The cost of the technology also needs to be acceptable to the potential users of the mesh. If it is too high the network may be under utilised. The CRCnet project has gained considerable experience in using small inexpensive computer hardware such as the Soekris[35] devices to accomplish this goal.

High Performance

The aim of the mesh network is to support broadband deployment to rural communities. Definitions of exactly what broadband is vary and often encompass features other than the speed of the connection, such as whether the connection is always on. The mesh network should be able to support the current generation of broadband applications as well providing the capacity to support future generations of broadband applications that may require even more bandwidth. Todays broadband applications include video conferencing, live video streaming, voice over IP and web browsing.

Ease of Use

The mesh network must require a minimum amount of configuration by end users. A rural household should be able to purchase a mesh node from a consumer electronics store, bring it home, bolt it to the roof and be connected to the mesh network. This requires the software on each node to be self-configuring, that is it must detect neighbouring nodes and establish wireless links without any user intervention. Additionally the software must also allow the network to be self-healing. When a node goes offline the remaining nodes must reestablish links to each other and maintain connectivity without any user interaction. A self-configuring, self-healing network would be very robust given a sufficient density of nodes.

Avoidance of Geographic Obstacles

New Zealand's rural environment presents many challenges to line of sight based wireless technologies. Obstacles such as trees and hills impede the passage of wireless signals between two nodes. The mesh network

should take advantage of all the nodes available to it in order to route around obstacles.

Directional Antenna

Directional antenna focus the signal transmitted from a wireless card in a single direction. This maximises the possible distance between two nodes and reduces the level of interference which will be very helpful in a rural environment. Section 2.2 describes directional antenna in more depth and expands on the rationale for their use in the proposed mesh network.

1.5 Project Aims

This project aims to contribute to the development of a platform to allow wide scale deployment of broadband Internet access in rural areas by designing and implementing a prototype wireless mesh network with the characteristics described above. This aim has been broken down into three smaller goals to be accomplished by this project.

- Investigate existing wireless mesh network protocols and implementations. Evaluate their performance, the challenges that they face and their suitability for use in a rural environment.
- Drawing on the lessons of existing protocols and implementations, design a mesh network protocol that can be used to provide a platform for wide scale rural broadband deployment.
- Develop a framework to allow the implementation of the designed mesh network protocol. Implement the protocol within the framework and evaluate its performance and suitability for use in a rural environment.

The project has been very successful in achieving these aims and the framework that was developed has been successfully used on a small rural mesh network.

1.6 Report Structure

This report consists of 7 major chapters. Chapter 2 provides an introduction to 802.11, its use in a rural wireless network and evaluates existing mesh networks built using 802.11b. This information is used to come up with some design guidelines for the mesh network to be designed by this project. Chapter 3 describes the protocol that has been designed to create the mesh network described in the end of Chapter 2.

Chapter 4 describes the development of the mesh framework that implements the protocol described in Chapter 3 and provides an environment for future development to occur. Chapters 5 and 6 present the results of this project alongside potential areas for further research before Chapter 7 summarises the report.

Chapter 2

Background

The previous chapter introduced the concept of using wireless technologies to provide broadband Internet access to rural areas. In particular wireless mesh networks were identified as a feasible technology to achieve this goal. This chapter describes in detail the specific wireless technology this project proposes to use and studies existing mesh networks that utilise it. The latter half of this chapter investigates current routing protocols designed for use with mesh networks and evaluates their suitability for use with the proposed mesh network.

2.1 Introduction to 802.11b

802.11b[5] is a technology standardised by the IEEE to provide a wireless equivalent to the wired Ethernet specification 802.3[6] which is typically used for local area networks (LANS). 802.11b operates in the Industrial, Scientific and Medical (ISM) spectrum band at 2.4Ghz. This band of the spectrum is licensed under a general user license described in Section 1.1.

2.1.1 802.11b Channels

The 2.4Ghz ISM band where 802.11b operates is 100MHz wide and is divided into fourteen overlapping channels as shown in Figure 2.1. Logically this allows fourteen individual networks to operate independently at any one time. However due to the overlap between channels interference is a problem and it is not feasible to use all fourteen channels at once. Regulations also affect the number of channels available for use by wireless networks. In many countries only the first eleven channels are able to be used. Of these eleven channels there are only three channels (1, 6 and 11) which do not overlap. Most wireless networks that are in use today use one of these three non overlapping channels. The increased usage of these three channels leads to interference problems due to the number of networks competing for access. In many urban areas of New Zealand 802.11b communications are extremely unreliable due to high levels of packet loss caused by interference.

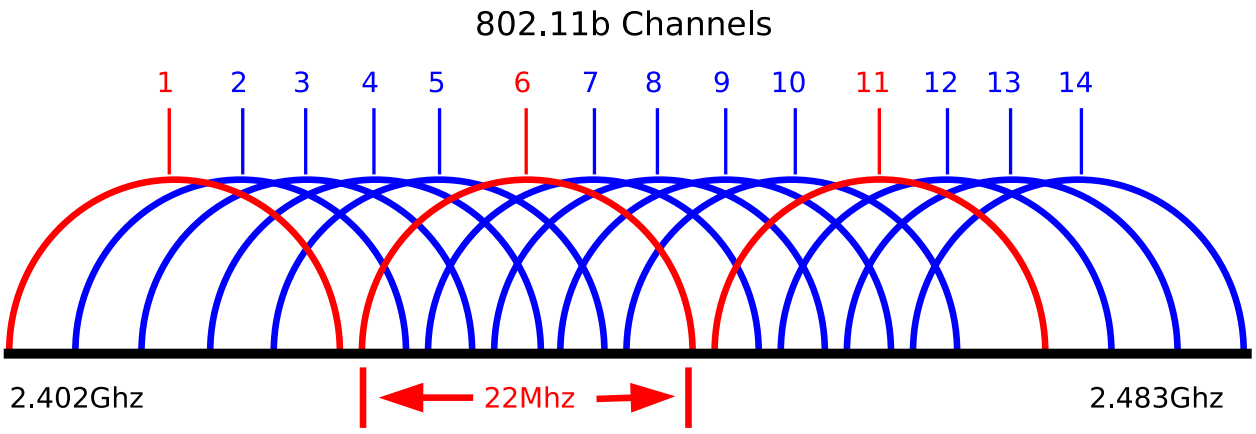

Figure 2.1: 802.11b Channel Assignments

2.1.2 802.11 MAC Layer

The purpose of the 802.11 MAC layer is to arbitrate access to the physical wireless medium and provide services such as addressing to support communications between upper level protocols. To ensure interoperability between existing wired Ethernet networks and 802.11 wireless networks the 802.11 MAC layer provides an interface that is identical to a wired Ethernet despite the fundamental differences between them. This design decision adds considerable complexity to the 802.11 MAC layer as it must implement features to make a wireless medium behave like a wired medium.

The fundamental difference between a wired and a wireless network is the ability to detect collisions. A collision occurs when two nodes in range of each other transmit simultaneously resulting in their transmissions colliding and being corrupted. Detecting collisions is important for the performance of a network protocol so that they can be recovered from and avoided in future. While it is easy to detect a collision on a wired network a wireless device is not able to detect a collision while it is transmitting for two reasons. First 802.11 radios are half-duplex, they cannot receive while transmitting and vice versa. Secondly, even if a full duplex 802.11 radio which could send and receive simultaneously was developed, the strength of the transmitted signal is so strong that any incoming transmission would not be heard. The 802.11 MAC layer tries to overcome the problem of not being able to detect collisions by requiring each packet to be positively acknowledged. If an acknowledgement is not received the packet is retransmitted as shown in Figure 2.2.

The 802.11 MAC layer is implemented in the control software of 802.11b radio cards. Although the 802.11 MAC layer is an open standard, the source code for the control software is regarded as proprietary property. Without access to this source code modifications to the 802.11 MAC layer are not possible.

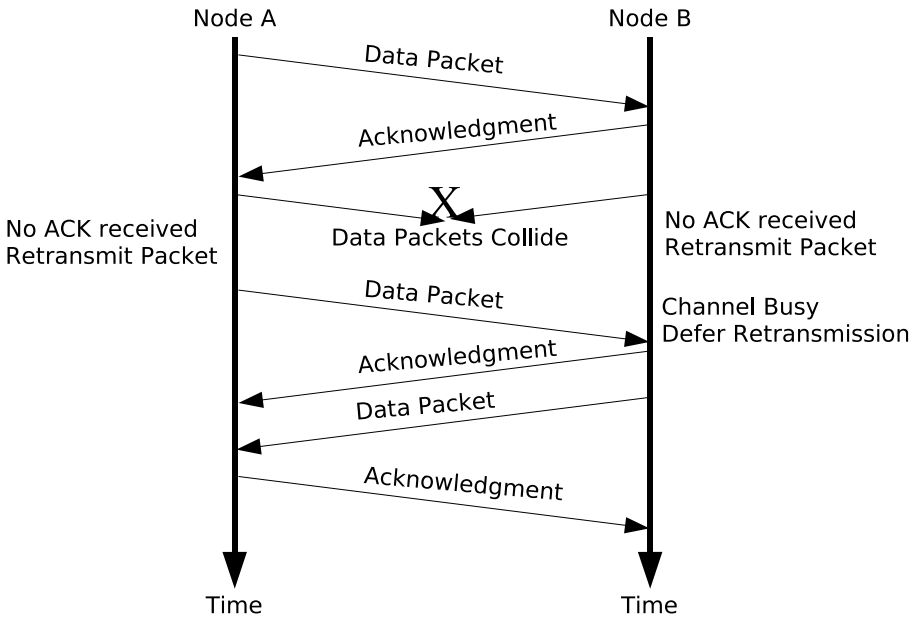

Figure 2.2: 802.11 Positive Acknowledgement

2.1.3 802.11 Network Modes

The 802.11 standard provides for operation in two modes, Managed or Ad-Hoc. These two modes affect how the MAC layer described in Section 2.1.2 interacts with other nodes.

Managed mode is only used when a client is connecting to a base station (called an access point in 802.11 terminology). Managed mode results in a broadcast network topology identical to that described in Section 1.2.1. All communications must pass through the access point and there can be no client to client communications. As access points are normally used to give wireless clients access to a traditional wired network the lack of direct client to client communication is not a large problem. In cases where two clients of an access point wish to communicate the packet must be sent to the access point which forwards it on to the destination client. This is extremely inefficient and takes twice as long as it would to send the packet directly between the two nodes. If the two nodes are too far from each other to communicate directly the use of an access point to relay their message is beneficial. However in the case where the two nodes are close enough for direct communication to occur the access point introduces unnecessary overhead.

Access points are particularly susceptible to contention based problems in the 802.11 MAC layer such as the hidden node problem[15]. This problem occurs when two client nodes that are not able to communicate directly attempt to transmit at the same time. As they are out of communication range they cannot detect that the other node is using the channel and mistakenly think it is free. When the two competing transmissions arrive at the access point they become corrupted and are retransmitted as described in the previous section. The 802.11 MAC layer contains a mechanism by which a node can attempt to reserve the channel before transmitting which is intended to solve this problem. Unfortunately the paper referenced above shows that

this mechanism is not effective at preventing collisions caused by the hidden node problem.

Ad-Hoc mode is used to connect two or more 802.11 clients in the absence of an access point. One common use for ad-hoc networks is to connect two nodes in a point to point link. Ad-Hoc mode can be used to connect more than two nodes however unlike access points if two nodes are out of range other nodes in an ad-hoc network will not relay packets for them. This problem means that ad-hoc networks consisting of more than two nodes need to implement some form of routing protocol to deliver packets between nodes. Point to point ad-hoc networks avoid many of the problems experienced in managed mode networks. Hidden nodes are not possible as by definition the two ends of the point to point link must be able to hear each other and contention for the wireless layer is also reduced as only two nodes are active on the link.

Unsurprisingly point to point 802.11 links provide much better performance than multi node links. Point to point links can be implemented in either ad-hoc or managed modes. The following section expands on the performance of 802.11 networks.

2.1.4 802.11b Performance

The advertised speed of 802.11b networks is 11Mbit/s, however this value is never achieved in real world operation due to protocol overheads. In addition some parts of 802.11 frames are transmitted at slower data rates to ensure compatibility with older versions of the protocol. Taking all these factors in to account the maximum achievable throughput of an 802.11b network running in ad-hoc mode is 6Mbit/s[14]. Collisions, interference and other protocol features can reduce this rate further.

The mode of an 802.11 network can also significantly affect the performance achieved. In the case of a managed network the 6Mbit/s figure quoted above cannot be achieved between two clients of an access point as each packet takes roughly twice as long to transmit as in an ad-hoc network. In this scenario the maximum achievable throughput between clients of an access point will be less than 4Mbit/s.

The level of performance offered by 802.11b is comparable to the current range of broadband technologies that are offered in suburban areas. A protocol built on top of the 802.11 MAC layer will also be easily adaptable to physical layers such as 802.11g that are able to provide much higher levels of throughput.

2.2 Antenna Technology

Most 802.11b cards have an internal antenna with a communication range of around 50m. These cards are designed for use in laptops and other mobile devices that are connected to a nearby access point. However it is possible to obtain 802.11b cards which allow external antenna to be connected to increase the communication range of the card. External antenna can be categorised into two broad groups.

Omni directional antenna distribute the power output by the wireless card in all directions simultaneously.

This allows a single wireless card to easily cover a large geographical area. However it increases the level of background noise that is received by the card and also the area in which the transmission causes interference to other wireless devices.

Directional antenna focus the power output by the wireless card into a single directed beam. This technique reduces the interference caused by the transmission as the signal is confined to the path between the transmitter and the receiver. Additionally because the power is focused in a single direction, rather than spread around all directions, the signal is able to travel further. This allows a directional antenna to increase the possible distance between two nodes.

Existing research into directional antenna has shown that their use can lead to an increase in the capacity of the network as several links can be operated simultaneously[17]. This result is supported by several other studies [32], [43]. Unfortunately this work is directed towards MAC layer protocol modifications which means it is not directly useful to this project due to the lack of access to wireless card control software.

2.3 Existing Mesh Networks

802.11b wireless cards and antenna have been available since late 1999 and there have been many community projects [34], [36], [40] aimed at providing wireless Internet services. While some of these projects loosely use the word ”mesh” in their descriptions the networks do not resemble a true mesh network (as shown in Figure 1.1) in any shape or form. This section describes two projects that aim to create networks with a topology that resembles a true mesh network.

2.3.1 MIT Roofnet

The MIT Roofnet[23] project has built a mesh network surrounding the MIT campus in Cambridge, Massachusetts, USA. The focus of the MIT Roofnet project is to develop routing protocols for use in mesh networks. Each MIT Roofnet node consists of a single 8dBi omni directional antenna connected to an 802.11b radio card operating in the 802.11 ad-hoc mode described in Section 2.1.3. The use of a single omni directional antenna for each node limits the entire network to a single channel. The software used to control each node is a combination of Perl scripts and kernel modules for the Click Modular Router[16] and runs on a modified version of the Redhat 9 Linux distribution. The software is freely available.

Documentation on the Roofnet network indicates that it performs well, with typical network throughput ranging from dozens to hundreds of kilobytes per second[1]. However as the entire network of around 50 nodes exists on a single channel within a dense area the aggregate throughput of the entire network will be much less than the 6Mbit/s figure from Section 2.1.4 due to collisions and interference between links. Roofnet is unlikely to be able to support high bandwidth applications such as video conferencing on any more than

two nodes in the mesh at any one time.

The Roofnet mesh network has been designed to have a very high density of nodes which results in a high number of possible links with a wide range of loss rates. This provides a rich environment for the development of routing protocols. The downside of having a very dense mesh network is the high level of interference caused between nodes.

The Roofnet project uses a routing algorithm called SrcRR to find the best path from each node in the network to a destination. SrcRR is based on the principles of the DSR[3] routing protocol but it has been modified to use a metric called ETX which is discussed further in Section 2.4.2.

While Roofnet is an interesting project it has significant differences from the mesh network required to fulfil the aims of this project. In particular, the density of nodes required by Roofnet to allow its routing algorithms to function is not able to be achieved in a rural environment where each node could be up to one or two kilometres apart, as opposed to the average distance between Roofnet nodes of around 100m. Roofnet's use of a single 802.11b channel for the entire network is also undesirable as it reduces the amount of bandwidth available to clients of the network.

2.3.2 Locust World

The Locust World[19] MeshAP is a Linux distribution designed to enable the rapid deployment of 802.11b networks. The MeshAP is sold commercially and while it is in active use today the only freely available versions are two years old. Little technical documentation on the topology of the network is available. Each mesh node uses a single antenna, in most cases an omni-directional antenna is chosen, but directional antennas may also be used. The mesh software places the wireless interface into managed mode to create an access point that other mesh nodes connect to. Locust World uses the AODV routing protocol described in Section 2.4.3 to find the best path through the network.

Locust World is a commercial project as opposed to an academic project. As such there is no publicly available data on the performance of the network. This fact and the lack of technical documentation on the architecture of the network make the Locust World MeshAP unsuitable as a base for the mesh network required by this project. Additionally the use of a single wireless interface for each node has the same drawbacks that were discussed with regards to Roofnet's use of a single wireless interface.

2.4 Routing in Mesh Networks

A key characteristic of the existing mesh network implementations studied in section 2.3 is the reliance on the routing layer to perform the critical tasks of neighbour discovery and route selection. Routing protocols for ad-hoc wireless networks are heavily oriented towards mobile networks (MANETS) and are optimised

for very dynamic networks. Wireless networks present an interesting challenge for routing protocols, which are traditionally based on a shortest path principle, as the shortest path is not always optimal. This section describes the challenge in more detail and presents a high level overview of several routing protocols that may be easily adaptable for use in the wireless mesh network this project aims to create. A more detailed evaluation of current mobile ad-hoc routing protocols can be found in [33].

2.4.1 Shortest Path is Not Enough

The job of any routing protocol is to select the "best" path through a network to deliver a packet to its destination. The routing protocols examined later in this section all use the concept of the shortest path[18] as the criteria for selecting a route. The shortest path is calculated by viewing the network as a graph where each link is an edge in the graph and each router is a vertex. Associated with each edge is a value that represents the cost (or metric) of using that link. Typically this cost is based on the bandwidth of the link.

Recent research has shown that traditional shortest path routing usually fails to select the optimum path for a packet through a wireless network[10] as the metrics do not take into account the varying quality of each wireless link. In particular as each wireless link in the network has the same bandwidth the metric is identical for each link and the routing protocol is forced to fall back to a simple link count to determine the optimum path. The number of links in a path provides no guarantee of quality on a wireless network. A path of 5 high quality links that have low loss rates should be preferred over a path of 3 low quality links with high loss rates. New metrics are needed to allow traditional shortest path routing algorithms to take into account the factors affecting the performance of wireless links.

2.4.2 Existing Wireless Metrics

Existing metrics used with shortest path algorithms are based on the link bandwidth which the previous section identified as being inappropriate for use on wireless networks. New metrics are needed which accurately represent the quality of wireless links. One possibility is to use factors such as the signal quality of a link or the physical distance covered by the link. Links with high signal qualities and covering short physical distances would be favoured over longer links or those with low signal quality. However recent observations conducted by the Roofnet project have found that neither of these factors are accurate predictors of the quality of a wireless link[2]. This observation suggests that a metric based on observed characteristics of the link would be best.

One such metric, also designed by the Roofnet project, is called ETX[9]. ETX works by regularly sending beacons and monitoring how many of the expected beacons are received over a specific time period. This statistic is used to infer a loss rate for the link, and consequently how many transmissions are expected to be needed to successfully transmit a wireless packet. This expected transmission count can be used to accurately

predict the performance of each link in the wireless network.

Any shortest path routing algorithm for which source code is available can be easily modified to use a metric such as ETX. An implementation of the SrcRR protocol which implements the ETX metric is available, however it operates on a single wireless interface only which is not suitable for this project.

2.4.3 Existing Routing Protocol Implementations

An overview of the many wireless routing protocols available can be found in [33], however it is not clear that any one protocol is better than the rest and should be used in the proposed mesh network. This section provides an description of several implementations that may be suitable and evaluates them in light of the problems with shortest path and existing metrics described in the previous section.

AODV

Ad-Hoc On-Demand Distance Vector (AODV)[27] routing is, as its name suggests, a distance vector routing protocol that calculates routes as they are required. When a route is needed the source node broadcasts a route request that is flooded through the network. When the request reaches the destination a response is sent back to the source node. During this process the routing table in any node traversed by the route request and route response packets is setup. AODV also utilises a periodic hello beacon to detect neighbouring nodes and monitor the quality of the link to them. One problem that has been identified with AODV is that the hello packets have significantly different characteristics to normal data packets. This leads to a situation where the routing protocol indicates that a valid link is available, but any packets sent over it experience a high level of packet loss[21]. A Linux implementation[25] of AODV is available, however it is not be able to be directly used with this project as it operates at the link layer and only supports nodes with a single wireless interface.

DSR and DSDV

The Dynamic Source Routing (DSR)[3] protocol is another on-demand routing protocol designed for ad-hoc wireless networks. The key feature of DSR is its ability to route packets in a network of nodes whose topology changes rapidly every few seconds. DSR is source routed, each packet sent into the network contains information on the hops it must take to reach the destination. This removes the need for intermediate routers to have full routing tables. Each node in a network utilising DSR maintains a cache of previous discovered routes. If the route cache does not contain a route for a destination DSR uses a route request / reply exchange similar to AODV to populate it. Once a route is in the route cache it is not removed unless it stops working. The AODV protocol does not specify how routes are monitored to ensure that they are working, but proposes several schemes such as watching for acknowledgements from high level transport layer protocols to achieve this.

The Highly Dynamic Destination-Sequenced Distance Vector (DSDV)[26] protocol is a link state routing protocol. Unlike AODV and DSR the DSDV protocol maintains a full routing table on each node in the wireless network at all times. This table is populated via hello messages sent by each node at regular intervals. When sending a hello message each node includes information regarding nodes it knows how to reach and how many hops it would take to reach them. In this way the routing table is propagated across the entire network.

A Linux implementation of the DSR and DSDV routing protocols is available through the Click Modular Router[16]. The Click implementations of DSR and DSDV have been modified to support additional routing metrics such as ETX described in Section 2.4.2. Both DSR and DSDV are able to be run at the link layer or the network layer. When running at the link layer only a single interface is supported.

OSPF

The Open Shortest Path First (OSPF)[24] routing protocol is designed for use on any IP network and has no specific features for use on wireless networks. Similar to AODV and DSDV, OSPF utilises hello messages to discover and maintain links with the direct neighbours of a node. In addition to the regular hello messages each node generates link state advertisements that are propagated to all nodes in the network whenever a link changes state. Each link in an OSPF network has an associated metric which is traditionally based on the bandwidth of the link. For each possible route to a destination a node calculates the sum of the link metrics in that path. The path with the lowest sum is chosen as the best route.

OSPF is a widely used routing protocol and has a mature Linux implementation as a part of the Quagga[30] routing suite. The Quagga routing suite is a set of programs that provide a variety of routing protocols to Linux and other Unix based operating systems. Quagga uses an interface layer between the kernel and the logic of the routing protocol to make it easy to modify and create new routing protocol implementations. The CRCnet project currently uses the Quagga OSPF implementation for the network described in Section 1.3.

2.5 Design Requirements

After reviewing existing 802.11b networks and available routing protocols the following design requirements were reached for the mesh network to be constructed by this project. These requirements also draw on the experience gained from building and managing CRCnet. Figure 2.3 shows the topology of an example mesh network similar to what this project would like to create.

The mesh network must ensure that every mesh node that is within communication range can communicate with all other nodes in the mesh. This communication must be achieved with a high level of performance and redundancy as described in section 1.4. Given the limitations of the 802.11 MAC protocol the mesh network

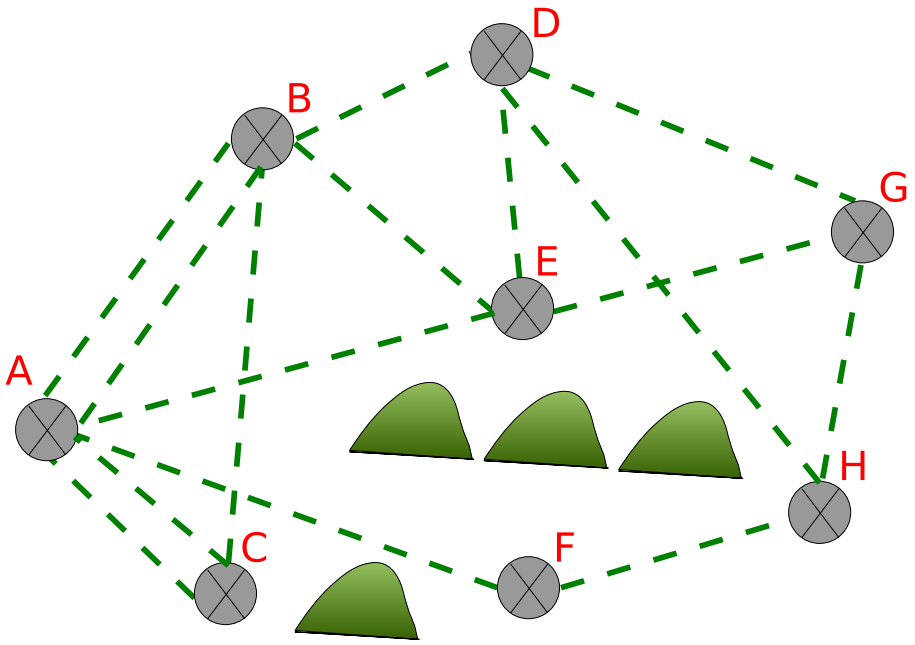

Figure 2.3: Example mesh network showing all possible links

should favour point to point links in ad-hoc mode to maximise performance. Some situations may require the use of point to multi point, or even multi point to multi point links to ensure that all nodes are connected to the network in a redundant measure. While this is acceptable to achieve the desired level of connectivity, point to point links should be used whenever possible.

The mesh network should utilise multiple wireless interfaces on each node, coupled with directional antenna to increase the overall capacity available to the network. The use of directional antenna also reduces the level of interference between links which allows denser networks to be constructed. A node with four wireless interfaces should be able to effectively communicate in all directions using a directional antenna on each interface. The use of multiple interfaces also increases the ability of each node to form point to point links with other nodes.

The implementation of the mesh network must be modular to allow for future development of the protocol without having to completely redesign it. For example while all development for this project will take place with 802.11b wireless cards, moving to 802.11g wireless cards should require only minimal changes to the implementation. The mesh network should also seek to reuse existing utilities and open source software such as the Linux kernel and the Quagga routing suite to lower development time as much as possible. Given the lack of access to the control software of the wireless cards the implementation must not require modifications to the 802.11b MAC layer.

Chapter 3

Mesh Protocol

This chapter describes the protocol that has been implemented by this project to create a mesh network with the characteristics described in Sections 1.4 and 2.5.

3.1 Protocol Aim

The aim of the mesh protocol is to take a set of independent nodes and form them into a mesh network. Each node in the network operates autonomously and the protocol must not rely on any form of centralised control. The protocol specifies the tasks each node must perform and how to perform them.

Each node in the mesh network must have a unique identifier, called the Node ID, and at least one wireless interface with which it can communicate with other mesh nodes. Multiple interfaces on a mesh node must be assigned an index to allow them to be uniquely identified in the mesh network.

3.2 Protocol Tasks

Creating a mesh network from a set of independent nodes requires each node to undertake four basic tasks, Neighbour Discovery, Neighbour Selection, Link Establishment and Routing. These tasks must be completed in order as each task relies on the results of the previous one. Tasks 1 and 3 are performed individually on each interface of a node, while tasks 2 and 4 operate on information gained from all interfaces of a node.

Each mesh node must discover if there are any other mesh nodes within its communication range before it can perform any other step. The process of finding other nodes within communication range is called neighbour discovery.

After performing neighbour discovery it is likely that a mesh node will have multiple choices in links to another mesh node. Neighbour selection involves choosing a subset of all possible links to be configured and become part of the mesh.

Once the node has selected the links to establish, the parameters of each link must be agreed upon with each neighbour node. Ensuring that each node is synchronised is crucial to the success of link establishment.

As the links in the mesh network are established, a routing protocol must calculate the best paths through the network and setup the routing table on each node.

The four tasks can be used to loosely describe the state of each interface on a node at any point in time. There is no requirement for all interfaces on a node to be in the same state at once, and an interface must be able to respond correctly to any packets that it receives regardless of it's current state.

3.3 Hello Protocol

Each of the tasks described in the previous section requires some level of information to be transfered or collected from neighbour nodes in the mesh network. Building on the techniques used in the AODV and OSPF routing protocols described in Section 2.4, a hello protocol is used for neighbour discovery, synchronisation and link maintenance in the mesh protocol.

3.3.1 Hello Packets

The operation of the hello protocol centres around the transmission of a hello packet every five seconds from each interface of a mesh node. Hello packets are always padded to 1500 bytes long to avoid the problems encountered in the AODV protocol described in Section 2.4.3. Additionally hello packets are directed to a specific neighbour node rather than being sent to the broadcast address to ensure that they are treated identically to normal data packets. In situations where an interface has no neighbour nodes (such as during neighbour discovery) the hello packet can be sent to the broadcast address.

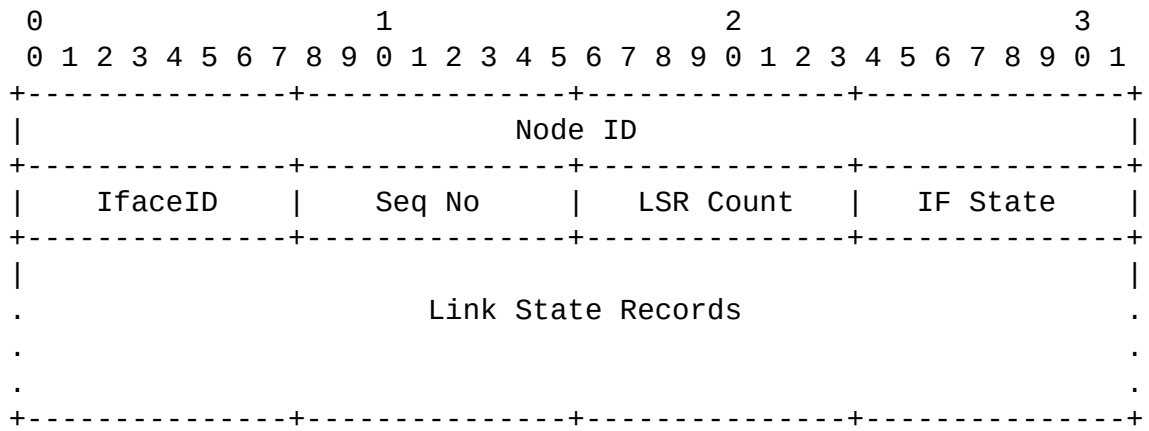

Figure 3.1: Hello Packet Format

Each hello packet contains the following pieces of information as shown in figure 3.1.

Node ID (4 bytes) – The unique identifier of the node that generated the hello packet .

IfaceID (1 byte) – The index of the interface on the node that generated the packet. Combined with the previous Node ID field this uniquely identifies where the hello packet came from.

Seq No (1 byte) – A sequence number that is incremented with every hello packet sent from the interface. This allows neighbour nodes to determine if a hello packet has been lost.

LSR Count (1 byte) – A count of the number of Link State Records (LSRs) contained in the second half of the hello packet.

IF State (1 byte) – The state of the interface that generated the hello packet at the time it was sent.

Link State Records (variable length) – A series of records of the format shown in Figure 3.2 describing each link in the sending nodes link database. The number of records in this section is specified by the LSR Count field.

The primary function of a hello packet is to allow the receiver node to determine that the sender node is still online. If a three consecutive hello packets are not received from a neighbour node it is assumed to be down and the interface will return to neighbour discovery mode.

The secondary function of the hello packets is to synchronise the link database on each node via the link state records contained in the second half of the hello packet.

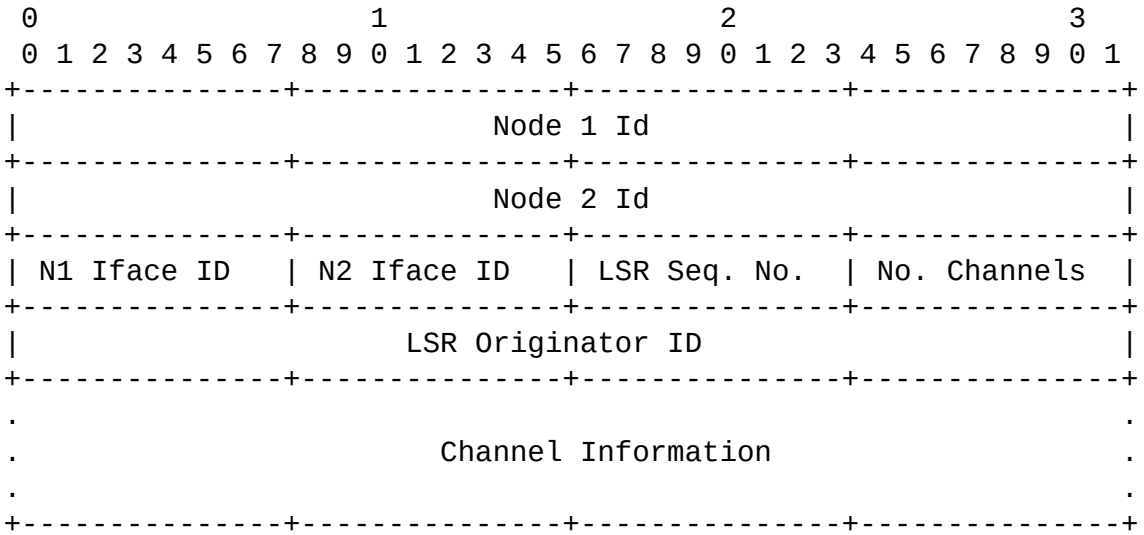

Figure 3.2: Link State Record Format

Each link state record contains the following pieces of information as shown in figure 3.2.

Node 1 ID (4 bytes) – The unique identifier for the first node that is part of the link.

Node 2 ID (4 bytes) – The unique identifier for the second node that is part of the link.

N1 Iface ID (1 byte) – The index of the interface on the first node that is part of the link.

N2 Iface ID (1 byte) – The index of the interface on the second node that is part of the link.

LSR Seq. No (1 byte) – A sequence number that is incremented every time a new LSR for this link is originated. This enables LSRs to be expired when the node originating them has left the mesh network.

No. Channels (1 byte) – The number of channel records contained in the second half of the packet.

LSR Originator ID (4 bytes) – The unique identifier for the node that originated this LSR.

Channel Information (variable length) – A series of records of the format shown in Figure 3.3 describing the channels that have been tested or are in use for this link.

Each link state record contains a number of channel information records that describe the quality of that channel, and its current state. Each field in a channel information record is described below and illustrated in Figure 3.3.

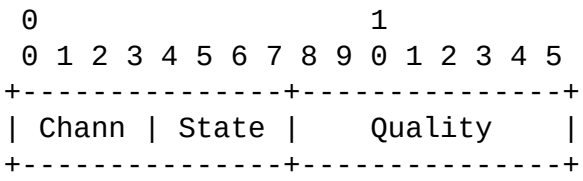

Figure 3.3: Channel Information Record Format

Chann (4 bits) – 802.11b channel number that this record relates to.

State (4 bits) – The current state of the channel. Valid values are:

2 (Active) – The channel is currently in use.

1 (Chosen) – The channel has been chosen for use but is not yet active.

0 (Available) – The channel is available for use.

Quality (1 byte) – The signal quality of the link on the specified channel as reported by the radio card. This value is updated with each hello packet that is received on the link.

3.4 Link Database

Each node in the network maintains a link database which describes all possible links between nodes in the network. The link database is populated with the information each interface reports about its directly

connected neighbours and from the link state records contained in received hello packets. When a new link entry is added to the link database it is given a timestamp. Each time a new link state record is received the originating node ID and the sequence number of the new record are checked against the database. If the originating node ID matches and the sequence number has increased the timestamp is updated, otherwise the new link record is discarded. Link entries related to directly connected neighbours are also regularly updated from the nodes interfaces. If the timestamp of an entry is not updated for seven consecutive hello intervals the entry is removed from the database.

3.5 Neighbour Discovery

The following procedure is carried out by a mesh node on each of its interfaces that is not currently part of an established link. If no neighbours are discovered when the procedure is initially carried out it is repeated every thirty seconds until a neighbour is discovered. In-between these repetitions of the procedure the node will transmit hello packets to the broadcast address so that any new nodes coming on line can hear the hello packet and respond.

3.5.1 802.11 Network Discovery

The 802.11 specification defines a procedure called a scan, in which the radio card very quickly monitors every channel and reports back a list of any active 802.11 networks that were found. This procedure is usually used by devices such as laptops to find access points, however there is no technical restriction on how it can be used. Performing a scan takes two to three seconds and returns a list of all 802.11 networks in the area. Other information such as the estimated signal strength of each network is also present in the response. This information provides an excellent first step in determining if there are any mesh nodes nearby, however it will also pick up 802.11 networks that are completely unrelated to the mesh. A filtering step to remove these unwanted results is needed.

3.5.2 Scan Result Filtering

To remove 802.11 networks that are not part of the mesh a two step filtering process is performed on the results of the 802.11 scan. Every link established by the mesh network has a string prepended to the start of its name. The first step in filtering the results of the 802.11 scan involves removing any network that does not begin with the correct string. After this step has been completed there is a fairly high chance that any remaining 802.11 networks belong to a mesh node, however a further check is needed to ensure that a laptop (or some other device) has not automatically associated itself with a mesh network node and is continuing to use the network name even after the mesh node has gone away.

The second filter step involves sending a probe packet onto each detected network. Any mesh node receiving a probe packet must immediately send a hello packet in response. If the node performing neighbour discovery does not receive a response to the first probe packet it will continue to send probe packets at five second intervals until a total of five probe packets have been sent. If no response is received in this time the node will assume that no valid mesh nodes exist on that network and discard it from the results. If a response to a probe is received the node updates the link database with the contents of the received hello packet and waits for the results of the neighbour selection algorithm described in the next section before proceeding further.

3.6 Neighbour Selection

After completing the neighbour discovery phase the mesh node must determine the desired topology of the mesh network in order to know how to configure each of its interfaces. The darker nodes and links in Figure 3.4 show the information that *node A* possesses about its neighbour nodes and the possible links that could be formed after completing the neighbour discovery step. In many cases there are several links that *node A* could form in order to communicate with a specific neighbour node.

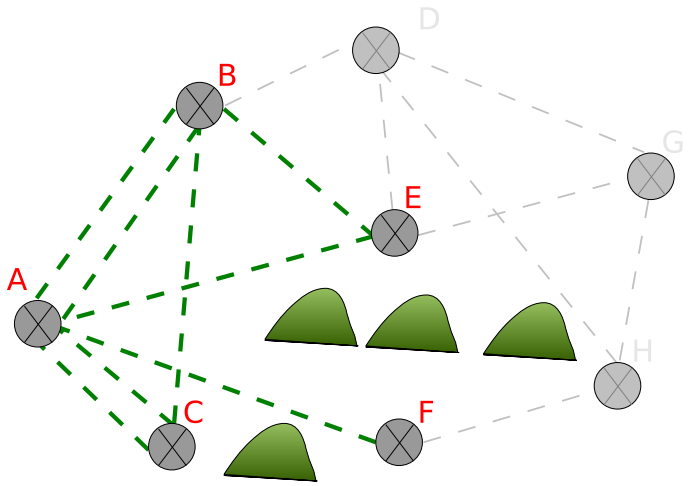

Figure 3.4: Node A's view of network after neighbour discovery

The task of the neighbour selection algorithm is to determine which links should be established and consequently the topology of the mesh network. This step is crucial to the performance of the mesh network as all further use of the network is constrained by the quality of the links that are chosen to be established in this step.

To maintain a reliable network provision should be made so that the failure of a single node can be accepted without taking any other nodes off line. This level of reliability can be achieved by making sure that there are multiple redundant paths between any two nodes. In the event of a node in the first path failing the traffic is simply routed over the second path.

Creating redundant paths through the mesh to increase reliability requires that mesh nodes be densely located so that many possible links are available for the neighbour selection algorithm to choose from. Unfortunately the closer the mesh nodes are to each other the higher the chance that their performance will be degraded by interference between them. This problem is aggravated when extra links are added to provide redundancy.

3.6.1 Spanning Tree Based Neighbour Selection

This project has implemented a neighbour selection algorithm that is based on the principle of a minimal spanning tree. The list of edges that can be used to build the spanning tree is built by extracting every link in the link database and computing a cost for each one. The cost of a link is calculated from the best signal quality that is available for that link and a factor based on the current state of the link. The factor favours links that are currently active in the mesh to provide stability. The list is sorted such that currently active links with the highest quality are first.

The spanning tree is built by iterating over the list of links in order and adding them to the tree until it contains *N-1* links, where *N* is the number of mesh nodes in the link database. This ensures that all nodes in the network are connected without providing any redundancy. When building the spanning tree only point to point links are used so that each interface on a node is part of a single link. If the end of the list of links is reached before *N-1* point to point links have been added to the tree the process is repeated from the start of the list with the point to point requirement modified so that a point to multi point link may be created if it connects a node that is currently isolated.

Figure 3.5 shows an example spanning tree network topology that could be created from the network shown in Figure 3.4 once all nodes have completed neighbour discovery. The spanning tree approach ensures that all mesh nodes are connected by high quality links fulfilling the two important design requirements of the mesh network.

3.6.2 Redundant Links

After the spanning tree has been constructed the algorithm performs three further iterations over the remaining links in the list. Each iteration reduces the quality that is required of a link in order for it to be added to the tree. The criteria applied to each link in the three iterations are shown in the following list.

1. Point to point links with high signal quality
2. Point to multi point links with high signal quality
3. Multi point to multi point links with high signal quality

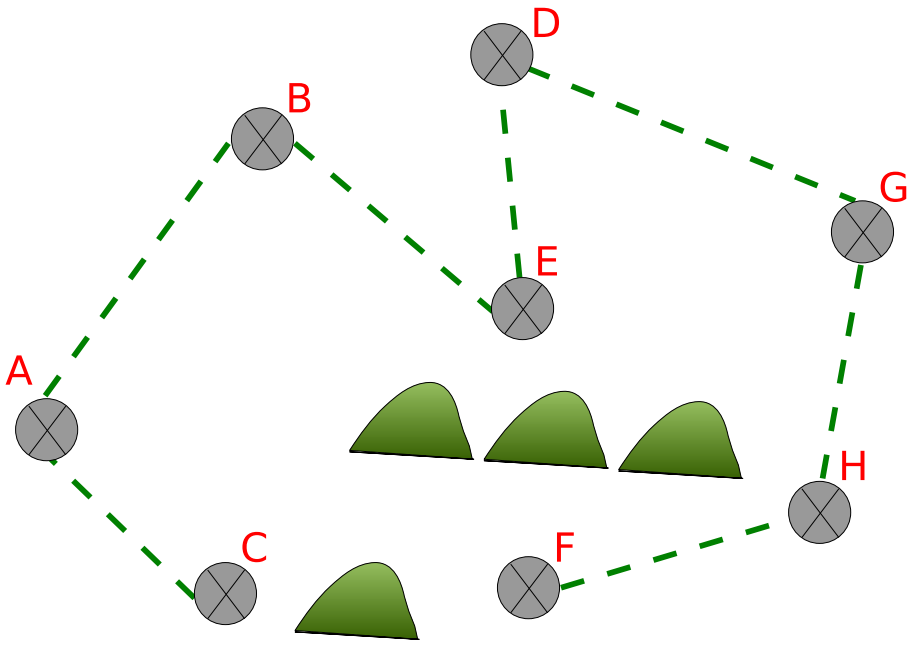

Figure 3.5: Example spanning tree network topology

This process transforms the spanning tree into a network graph that can be presented to the rest of the mesh protocol and ensures that the redundant links added to the mesh are of the highest possible quality. The algorithm stops at the end of the three iterations or when one of the following two conditions is met.

1. Every interface that has neighbours is part of a link, or
2. Each node has a link to at least two different neighbours

After completing this step the spanning tree network shown in Figure 3.5 will have been transformed into a complete mesh network as shown in Figure 3.6.

3.6.3 Channel Allocation

Each link that is selected to be part of the mesh network is assigned a channel by the neighbour selection algorithm. The algorithm implemented by this project uses a random scheme to allocate channels. The algorithm tries to ensure that each node does not use the same channel on more than one interface to avoid interference. The algorithm utilises all fourteen 802.11b channels but has no knowledge of which channels overlap. An attempt is made to use the channel that the link was initially discovered on but this may not be possible if two interfaces on a node discovered links on the same channel.

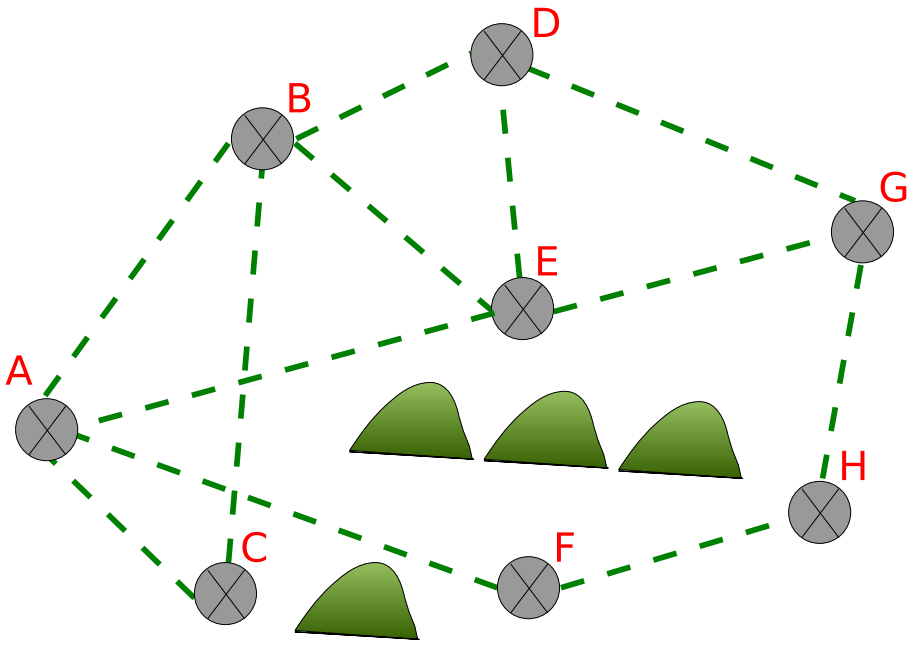

Figure 3.6: Example mesh network created by a neighbour selection algorithm

3.7 Link Establishment

After the neighbour selection process has been completed the node must ensure that each of its interfaces is configured to participate in the appropriate links. A crucial step in configuring each interface is to ensure that the neighbouring mesh nodes that the link is to be established with have a synchronised view of the network. If nodes are not synchronised their neighbour selection algorithms will form different network topologies and the link may not be established.

3.7.1 Neighbour Synchronisation

Neighbour synchronisation is achieved through the use of hello packets as described in Section 3.3.1 and a simple handshaking procedure. Figure 3.7 shows the series of packets that are exchanged between *Node A* and *Node B* as they establish a link. The top of the diagram shows each node at the completion of the neighbour selection algorithm but before a link is established. By the bottom of the diagram the nodes have established a link and are exchanging hello packets at regular intervals.

Node A initialises the link by sending a hello packet directly followed by an invite packet to *Node B*. The invite packet contains the the ESSID, 802.11 channel, 802.11 mode and IP network that *Node A* believes should be used for the proposed link. When *Node B* receives the hello and invite packets they are processed immediately. After checking the parameters contained in the invite packet match its configuration *Node B* sends an accept packet back to *Node A* and configures the interface with the parameters that were agreed upon. When *Node A* receives the accept packet it configures the interface with the agreed parameters. At this

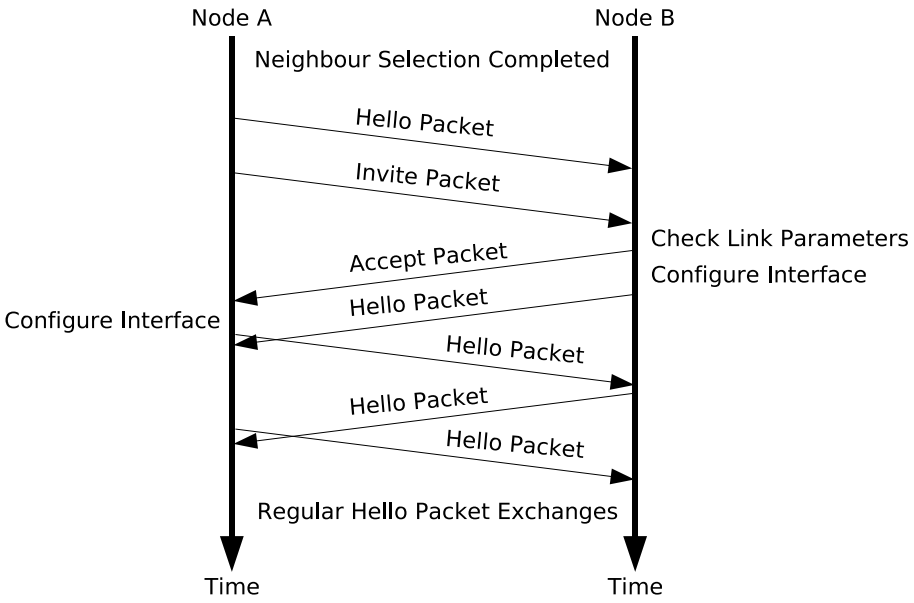

Figure 3.7: Neighbour Synchronisation Procedure

point the link is established and both nodes commence the regular hello packet exchange.

If the link databases of *Node A* and *Node B* are not synchronised they will not agree on the parameters of the link (or even that the link should exist). In this scenario when *Node B* receives the invite packet it will send only a hello packet back to *Node A* without sending an accept packet. This hello packet combined with the hello packet that *Node A* sent to *Node B* before the initial invite packet serves to synchronise the link databases of the two nodes.

If *Node A* does not receive any response to its initial invite packet it will retransmit the invite every five seconds until it has been sent a total of five times. If a response still has not been received *Node A* assumes that the link is no longer possible and returns to neighbour discovery mode.

3.7.2 IP Addressing

IP addresses in the mesh network are always allocated to point to point links between two nodes. If an interface is part of a point to multi point link, or a multi point to multi point link where it has multiple neighbour nodes several IP networks are allocated and the interface ends up with several IP addresses.

The benefit of this scheme is that the logic for assigning and configuring IP addresses on an interface is the same, regardless of the type of link that the interface is participating in. One IP address is always added to the interface for each directly connected neighbour node. Additionally using only point to point IP networks solves a problem encountered in point to multi point and multi point to multi point links where the nodes are not connected in a full mesh. By configuring point to point links only between those nodes that can directly communicate (as established by the hello protocol) packets travelling between nodes that cannot directly communicate must be routed over the point to point links.

For this project each node in the mesh network has a pool of IP addresses that it 'owns' and may not be used by anyone else on the network. When a node generates an invite packet to send to a neighbour it allocates an unused network from within its pool to the link and includes the details in the invite packet. The node receiving the invite packet does not allocate any IP addresses from its pool as it uses the network specified in the invite packet.

Each link is allocated an IP network with a CIDR[39] prefix length of thirty bits. This leaves two bits of the IP address to identify hosts on the network, allowing four possible addresses, however the first and last address in each network are reserved for special purposes, leaving two IP addresses available to identify hosts. The node with the lowest node ID gets the lowest address. Each node is given a pool with a CIDR prefix length of twenty four bits, this pool contains sixty four networks with a prefix length of thirty bits which allows each interface on a node to participate in sixteen point to point links. Given the mesh protocols preference for point to point links it is hard to envisage this limit ever being reached on all interfaces of a node at one time. Currently the mesh protocol automatically allocates a pool of IP addresses to each node based on its node ID.

3.8 Routing

At this point in the process a network such as the one shown in Figure 3.6 has been constructed. Each node in the network is able to communicate with its immediate neighbours, however it has no idea where to send packets for nodes that are not direct neighbours. Imagine that *Node A* in the figure has a packet that it wishes to send to *Node G*. Should the packet be sent to *Node B*, *Node C*, *Node E* or *Node F*? The job of the routing protocol is to discover all possible paths between any two nodes in the network, select the best performing path to use for delivering packets to their destination and ensure that the appropriate records are maintained on each node such that it can quickly determine which neighbour to send a packet to.

The mesh network constructed by this project consists of many wireless Ethernet links connected via a series of routers. A network of this type is not able to use the current implementations of specialised wireless routing protocols such as AODV, DSDV and DSR described in Section 2.4.3 as they operate on a single Ethernet link only. However the network is perfectly suited to use an IP layer routing protocol such as OSPF. This project uses the OSPF daemon from the Quagga routing suite. Currently the routing daemon is completely separate from the implementation of the mesh protocol and does not interact directly with it. The standard OSPF bandwidth based link metric is used which is not ideal given the shortest path problems identified in Section 2.4.1.

Chapter 4

Implementation Framework

The final phase of this project was to construct a framework that implements the protocol described in the previous section. The emphasis of this framework is on modularity such that elements of the mesh protocol can be easily replaced in the future as more sophisticated techniques are developed. This section describes the architecture of the framework and how it has been implemented.

4.1 Implementation Environment

A typical environment that the mesh network framework must operate in is the CRCnet Biscuit PC Linux Distribution. This is a customised Linux distribution built as part of the CRCnet project that operates on small computers commonly called “Biscuit PCs”. These Biscuit PCs typically have a 486 processor with 64MB of RAM. There is no writable hard disk available, a single read only 64MB compact flash card is used to store the operating system and applications. Due to the limited amount of space many common features are not available.

The Biscuit PCs have two PCMCIA slots and a single miniPCI slot allowing up to three radio cards to be used at one time. Solutions are currently being investigated to allow the desired number of four radio cards to be operated at one time. For the purposes of this project all mesh nodes were configured with a maximum of three wireless interfaces.

4.2 Implementation Language

One of the earliest decisions that had to be made regarding the implementation was the programming language to use. The choice of a low level language such as C would make the network and wireless configuration parts of the framework much easier, primarily due to the kernel and related tools also being written in C allowing a tight level of integration. On the other hand using a high level language such as Java or Python would allow

the more logic based parts of the framework to be written (and rewritten) quickly, where C would slow down development.

The emphasis of this project on creating a framework to enable further development of the mesh protocol in the future led to the decision that the rapid model of development offered by a high level programming language outweighed any benefits that a low level language would offer. The CRCnet Biscuit PC distribution did not contain any high level programming languages at the start of this project and one of the initial hurdles that was overcome was determining which language to use and installing it on the Biscuit PCs.

The Java Runtime Environment requires a minimum of 75MB of free disk space and recommends that you have 48MB of RAM available[13]. Neither of these requirements can be adequately met with the limited resources available on a Biscuit PC. Java is not open source, so the opportunity to recompile it leaving out unneeded features to save space is not possible. Python, as an open source application, is able to built from source and in this way a standard Python installation was able to be installed on a CRCnet Biscuit PC using less than 5MB of disk space.

Python has proved to be an excellent choice of language for the framework development. Had a low level language such as C been used for the mesh framework it is unlikely that this project would have progressed as far as it has towards creating the desired mesh network. One example of a benefit that Python has brought to the development process is its wide library of tools and functions. During preparation of the presentation for this project a need was identified to have real-time visualisations of the network topology. This presented the problem of how to get the data from each mesh node. Using one of Pythons built in modules a web server that returned a single page describing the state of the network was built in under thirty minutes. To implement a web server and integrate it with the existing code base would have taken at least a day in a lower level language such as C.

Despite the large number of modules that are present in the standard Python distribution there were some notable tasks a mesh node needs to perform that Python is not able to do. Fortunately Python has a very strong modular architecture allowing small snippets of C code to be easily inserting into the Python framework to extend its functionality. These user supplied modules can then be utilised in the same way as any built in Python module. Two such modules were written in the course of this project to provide functionality missing from Python that was required for the implementation of the mesh framework. Both of these modules will be released to the open source community in the near future.

4.2.1 Python Netlink Module

The standard Python distribution does not provide any way for a Python program to manipulate the network interfaces of the computer it is running on. A mesh node needs to be able to bring interfaces up and down at various times of its operation as well as being able to add and remove IP addresses from interfaces. To solve

this problem a python module was written in C to provide additional functionality to the mesh framework.

The interface exposed to python scripts by the netlink module is documented in Appendix A.

4.2.2 Python Wireless Tools Module

The Linux kernel includes a driver abstraction layer to allow standardised configuration of a range of different wireless hardware. This abstraction layer is called the Linux wireless extensions and is used in conjunction with a set of userspace tools[42]. Python does not provide a standard module to interface with the Linux wireless extensions.

The module that was written in the course of this project provides a generic interface to the Linux wireless extensions by reusing parts of the Linux wireless tools source code in a Python C module. The module allows Python programs to configure the important parameters for a wireless interface such as the ESSID, channel and mode. Additionally the module provides access to the scanning functionality of the radio card which is crucial to the success of the neighbour discovery procedure described in Section 3.5. Development of this module encountered several problems with data structures inside the wireless extensions being changed during the course of the project. This usually resulted in strange bugs such as the network name missing the first two characters. Careful effort needs to be put into monitoring the version of the wireless extensions that are being used as they have very a loose versioning scheme.

The use of the Linux wireless extensions layer to provide access to the underlying hardware is desirable as it accomplishes most of the work that is needed for a seamless transition between 802.11b wireless cards and 802.11g wireless cards.

The interface exposed to python scripts by the wireless tools module is documented in Appendix B.

4.3 Mesh Framework Architecture

The mesh framework that has been implemented by this project draws components from several different places. The key component of the framework is the mesh daemon which implements most of the protocol logic. This daemon interacts with the Linux kernel via several python modules including the two written by this project that were described in the previous section. Figure 4.1 shows how the various components of the framework fit together. Components with a dashed border are existing projects that were not written by this project.

4.3.1 Mesh Network Daemon

The key component of the mesh framework is the mesh daemon (shown in the top left hand corner of Figure 4.1). Within the mesh daemon are two main code units. These are implemented as separate threads and are

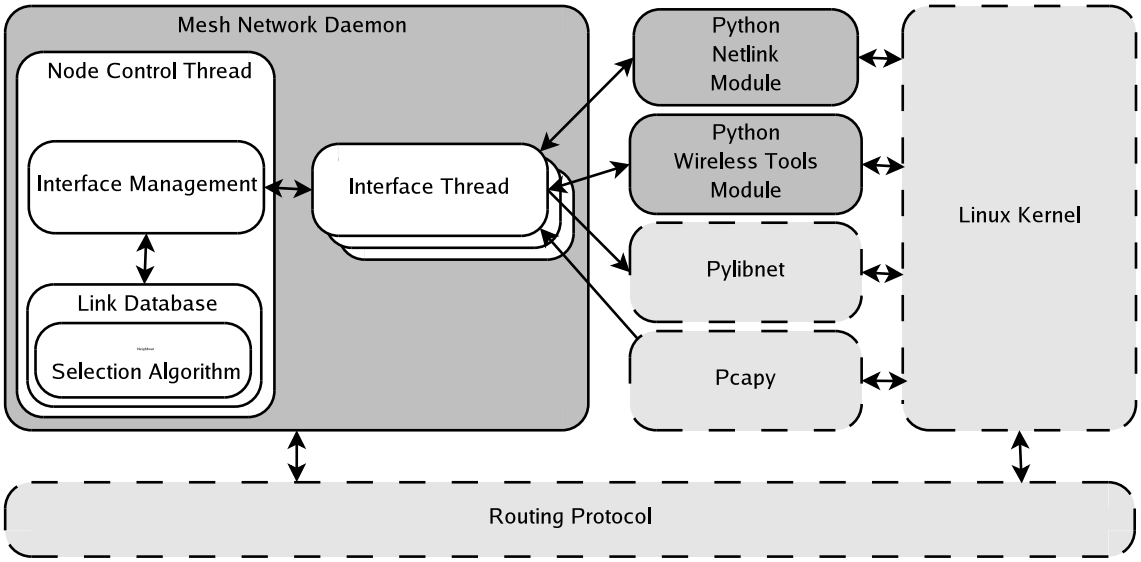

Figure 4.1: Mesh framework architecture

unshaded in the diagram. The first unit is the node control thread which controls the operation of the entire mesh daemon and ties the other units together. Only one node control thread exists in each instance of the mesh daemon. The second unit is a per interface thread created by the interface management class in the node control thread as part of the node initialisation.

The interface threads communicate with each other and the rest of the mesh daemon through the interface management class. The interface management class continually polls each interface thread to retrieve updated information on the directly connected neighbours which can be used to update the link database. Every five seconds the node control thread executes the neighbour selection algorithm. If this process changes the configuration for an interface, the interface management class is responsible for informing the appropriate interface thread of its new configuration. The structure of each interface thread and the tasks it performs is described further in Section 4.4.

The node control thread also contains the link database and neighbour selection algorithm. These components have been implemented as separate classes to allow for new implementations to be easily inserted in the future.

4.3.2 Routing Protocol

This project utilises the Quagga routing suite introduced in Section 2.4.3 to perform routing on the network using the OSPF routing protocol. The Quagga suite is run entirely independently of the mesh daemon on each Biscuit PC and has a configuration that enables it to detect and utilise links as they are established and removed by the mesh daemon. There is no direct communication between the mesh daemon and Quagga as Quagga is able to receive messages from the Linux kernel to automatically notify it when an interface on the

node is modified. An example Quagga configuration is provided in Appendix C.

While integrating Quagga into the mesh framework a bug was observed that prevented Quagga from detecting changes to an interface if they occurred after Quagga had started running. This was a serious problem as it meant that every time the mesh framework reconfigured an interface on a node there was the possibility that Quagga would fail to detect the change and routing on that interface would be disabled. This in turn could lead to a node being unable to fully participate in the network as each interface loses its routing over time. In conjunction with the Quagga maintainers the bug was diagnosed and a patch to Quagga was written and tested in the mesh framework. This patch has been submitted to the Quagga maintainers for integration into a future release of Quagga[31]. Further details on the diagnosis and fixing of the bug can be found in the Quagga mailing list archives[29].

4.4 Interface Threads

The interface thread is a critical part of the mesh daemon described in Section 4.3.1 as it is responsible for all packet transmission and processing tasks as well as ensuring that the physical interface on the node has been given the correct configuration. To implement the mesh protocol designed by this project each interface transmits and receives mesh protocol packets at the Ethernet level as opposed to the IP level. This approach is taken as it allows nodes to communicate as soon as the 802.11 parameters for the network have been configured without having to configure IP addresses. Transmitting and receiving packets at the Ethernet layer raises several problems as the standard Python libraries only support sending and receiving packets through the standard Linux IP stack.

Two existing Python modules were able to be used to solve this problem. The first module, named Pcapy[7] provides an interface to the Linux PCAP[37] library which provides functions to capture all packets received at the Ethernet layer of an interface and forward them to a userspace application (such as the mesh daemon) for processing. The second module, Pylibnet is a Python wrapper for the well known libnet library[38] and allows Python programs to generate arbitrary Ethernet frames to be transmitted by the interface. Together these modules allow each interface thread on a node to transmit and receive the mesh protocol packets.

Every mesh protocol packet sent from an interface has the header shown in figure 4.2 prepended to it to allow mesh protocol packets to be distinguished from standard data packets on the wireless link. The header also contains a simple checksum over the packet to ensure that the information received by the remote node is the same as what was originally sent. Each field in the packet is described below.

Version (1 byte) – The mesh protocol version that the format of this packet adheres to. If the version is greater than the version the receiving node implements the packet must be discarded.

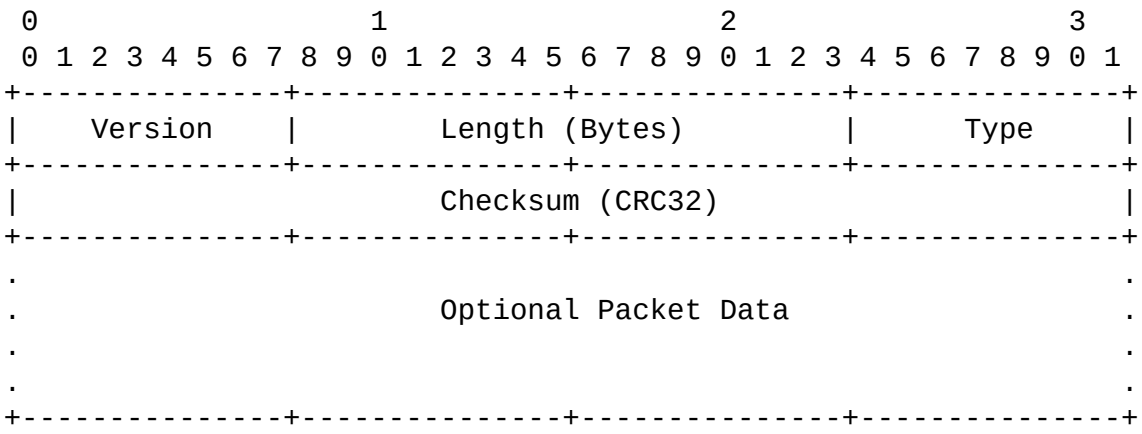

Figure 4.2: Mesh Framework Packet Header

Length (2 bytes) – The length in bytes of this packet, including the header.

Type (1 byte) – The type of mesh protocol packet. Valid values are:

- **0** (Probe) – Probe packets used for neighbour discovery.
- **1** (Hello) – Hello packets as described in Section 3.3.1.
- **2** (Invite) – Invite packets used for neighbour synchronisation as described in Section 3.7.1.
- **3** (Accept) – Accept packets used for neighbour synchronisation as described in Section 3.7.1.

Checksum (4 bytes) – A CRC32 checksum of the entire packet (including header). For the purposes of calculating the checksum the checksum field is set to all zeros.

Each interface thread contains a lock which is set whenever a portion of code is modifying the interfaces internal data structures. This lock is provided by the Python thread class and prevents the interface thread data structures from being corrupted as concurrent modifications are blocked by the lock.

Chapter 5

Results

The mesh framework created by this project was evaluated on two test bed networks. The primary characteristic that was evaluated in each case was the ability of the framework to establish wireless links through the network and setup routing. Performance measurements of the resulting networks have not been performed at this time, however as the networks that are created by this framework simply consist of a series of ad-hoc point to point links as used by CRCnet there is no reason to believe that the performance of the mesh network will be significantly different to that of CRCnet. Both test-beds were built using Soekris Biscuit PCs running the CRCnet Biscuit PC distribution and the mesh framework.

5.1 Indoor Mesh Test Bed

The first of the test bed networks was built inside the WAND Lab in G1.02 and consisted of three mesh nodes arranged in a triangular topology as shown in figure 5.1.

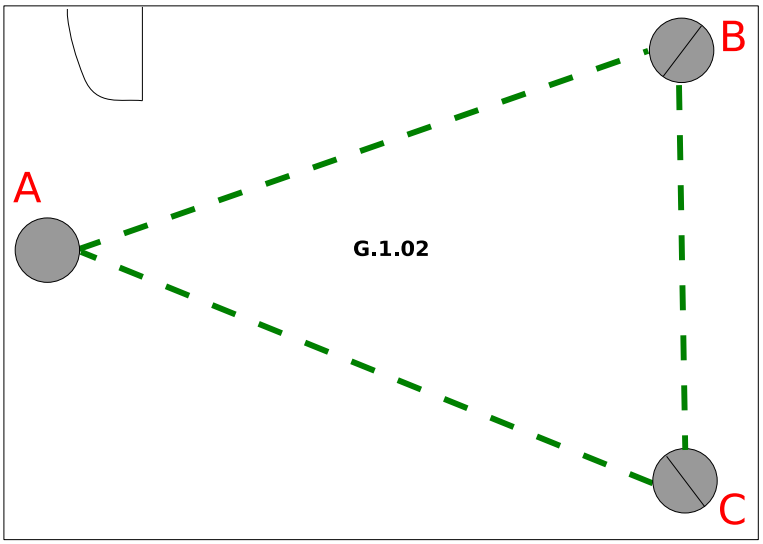

Figure 5.1: Topology and possible links in the G1.02 test bed network

This network topology gives three possible logical links between the nodes and would require two wireless cards in each node to establish each link on a separate physical channel. Although there were no technical limitations preventing the network from being operated with two cards in each node, most experiments conducted on the network focused on testing point to multi point links and multi point to multi point links. To achieve this *Node A* was configured with a single wireless interface.

The mesh framework operates correctly in all possible configurations of this mesh test bed, right from the scenario where each node has a single wireless card and a single multi point to multi point link is created to the ideal situation where each node has two wireless card and three completely independent links are created. The framework also correctly configures routing and full connectivity is achieved in all configurations.

The major problem identified with this testbed was due to the proximity of the nodes. The signal strength values reported by the wireless card for each neighbour were so strong that there was no way to determine which link was best. This problem can be easily solved by testing the framework in a more realistic environment.

Locating the test bed inside an office building with lots of 802.11 activity also provided an excellent test for the filtering scheme of the neighbour discovery algorithm. Each node in the test bed was able to hear at least three or four other unrelated 802.11 networks during the span of this project and several times devices that were not mesh nodes were discovered using a network name that began with the special string prepended to mesh network names. The mesh framework correctly discarded all these results.

5.2 Outdoor Mesh Test Bed

The second test bed network was built in the Ngahinapouri countryside to the west of Hamilton. This location was chosen as it is a typical rural environment with obstacles such as trees, hills and buildings that the mesh network must route around. Additionally it is close to CRCnet which means the network is easy to access.

Five nodes were installed on houses and a milking shed and statically configured 802.11b links were setup so that the network could be checked before the mesh framework was tested. This step did not go as smoothly as was planned. Although the CRCnet project has a large amount of experience in configuring rural 802.11b point to point links, they have mostly been from hilltop to hilltop rather than from roof to roof. On average it took twice as long to setup the links in this testbed network as it would to setup a normal point to point link on CRCnet. This extra time was spent engineering each link so that it could operate successfully despite the trees that obscured the line of sight between each node. This suggests that a high density of nodes will be needed in order to form rooftop level links in a rural environment.

Unfortunately due to the rural environment and the problems with link quality described in the previous paragraph there are a very limited number of viable links in the mesh network. This does not provide a chal-

lenge for the neighbour selection algorithm portion of the mesh framework as there is only a single choice of link in all cases. However the use of this test bed was extremely useful in verifying the synchronisation features of the mesh protocol. The mesh framework operates correctly on this test bed and provides connectivity to all nodes in the network. At the time this report was written this test bed was operational and provides Internet access to three rural houses including the house of an Associate Professor in the Computer Science Department. Figure 5.2 shows the topology of the network. This diagram was automatically generated by *Node 1* and shows the links between each interface on a node. Beside each link is the network name, the channel and the current state of the link.

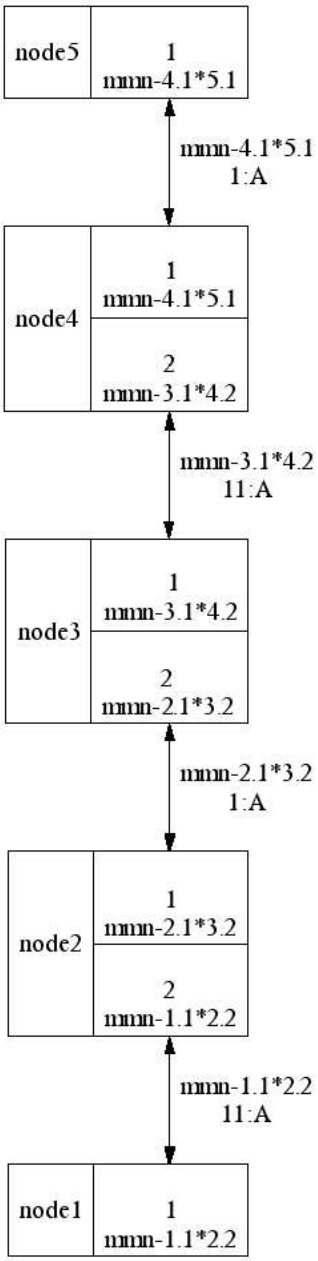

Figure 5.2: Topology of outdoor test bed network generated by FUL node

Chapter 6

Future Work

There are multiple areas that work on this project can continue from, the most obvious of which is to enhance the existing mesh protocol by developing more sophisticated algorithms for the allocation of channels and IP addresses. Currently the schemes used for this are very inefficient and in the case of channel allocation has the potential to create interference that leads to bad performance.

The protocol should also be extended to allow link state records to extend over two hello packets in the situation where the network is so large that they cannot fit into a single packet. This limitation has not been a problem for this project, however assuming each link record contains information on two channels there is only space for 73 link records in a single hello packet. The limit of what can be carried in a single hello packet is likely to be reached at networks of around seven or eight nodes depending on their density and the number of possible links between them.

This project utilises separate algorithms for establishing wireless links and routing packets over those links. However there is a significant overlap in the tasks performed by these algorithms. Both utilise a hello packet exchange to maintain neighbours and both need to propagate information on the state of links throughout the network. This overlap suggests that the two layers could be easily merged. Further research is needed to determine the ways in which this could occur and the possible impact on performance from merging the link and routing layers.

Currently the mesh protocol always favours point to point links regardless of the location in the network. Advanced schemes such as requiring point to point links in the centre of the mesh but allowing more relaxed link types such as point to multi point towards the edges could be investigated. There is also potential for integrating research into improving the performance of the 802.11 protocol to be integrated into the mesh network.

While this project did not have access to the control software of the wireless cards, this could change in the future. Efforts are underway within the WAND Network Research Group to develop a wireless card[41]. This wireless card would be very flexible and could possibly run an embedded version of Linux. The ability

to embed the mesh framework directly onto a wireless card is very desirable. Unfortunately while Python is able to operate effectively on the CRCnet Biscuit PCs it would be unable to operate in the more restricted environment offered by a wireless card. This problem could be solved by re-implementing the mesh framework in a lower level language such as C.

Chapter 7

Conclusion

The aim of this project was to contribute to the development of a platform to allow wide scale deployment of broadband Internet access in rural areas through the development of a wireless mesh network. This project has made considerable achievements in the three goals that were set.

The review of existing wireless technologies, mesh networks and routing algorithms revealed problems with existing implementations that prevented their use in rural areas.

Taking into account these problems and experience gained from the CRCnet project a protocol has been designed. This protocol creates a mesh network from a set of wireless nodes and is intended for use in rural areas. A distinguishing feature of this mesh network is the use of multiple wireless interfaces, each attached to a directional antenna.

The mesh protocol has been implemented inside a framework developed by this project. This framework is modular to allow different implementations for parts of the project to be quickly inserted. This framework, and the mesh protocol, have been successfully used on two test bed mesh networks. The success of the protocol on these two test bed networks demonstrates that the fundamental concepts behind the design of the mesh network are correct. The protocol met the requirement of self-configuration with ease and the topology of the network was created with no human intervention.

This project has achieved its aim of contributing to the development of platform to allow wide scale deployment of broadband Internet access in rural areas. This aim has been achieved by the development of a mesh protocol and a framework to allow future development. The potential for further research based on this project is large.

Appendix A
Pynetlink Module Interface

The Python module written by this project to provide access to the interface manipulation functions of the Linux kernel is called *Pynetlink* and can be included in any Python program with the command shown below.

```
import pynetlink
```

The *Pynetlink* module provides the following methods to Python programs. If an error occurs during the execution of a method an exception is raised and the method will return *NULL*.

get_iflist() Returns a list of the network interfaces on the node.

get_flags(ifname) Returns a integer representing the flags the are currently set for the interface specified by the *ifname* parameter.

set_flags(ifname, newvalue) Sets the flags value for the interface specified by the *ifname* parameter to the value specified by *newvalue*.

get_flag(ifname, flag) Returns either the *FLAG_ON* or *FLAG_OFF* constant to indicated the state of the flag specified by the *flag* parameter on the interface specified by the *ifname* parameter.

set_flag(ifname, flag, newvalue) Sets the value of the flag specified by the *flag* parameter on the interface specified by the *ifname* parameter to the value in the *newvalue* parameter which must be either *FLAG_ON* or *FLAG_OFF*.

flush_ifaddr(ifname) Removes all IP addresses from the interface specified by the *ifname* parameter.

get_ifaddr(ifname) Returns a list of the IP addresses assigned to the interface specified by the *ifname* parameter.

get_ifaddr(ifname) Returns a list of the IP addresses assigned to the interface specified by the *ifname* parameter. Each entry in the list will be an IP address attached to a netmask in CIDR format.

add_ifaddr(ifname, address [, brd_addr)] Adds the IP address specified by the *address* parameter to the interface specified by the *ifname* parameter. The IP address must be specified in CIDR format. An optional broadcast address for the IP address may also be specified.

del_ifaddr(ifname, address [, brd_addr)] Removes the IP address specified by the *address* parameter to the interface specified by the *ifname* parameter. The IP address must be specified in CIDR format.

In addition to the standard constants defined in the *net/if.h* header the *Pynetlink* module also provides two new constants intended for use when manipulating interface flags.

- FLAG_ON
- FLAG_OFF

Appendix B
Pyiwtools Module Interface

The Python module written by this project to provide access to the wireless extension functionality in the Linux kernel is called *iwtools* and can be included in any Python program with the command shown below.

```
import iwtools
```

The *iwtools* module provides an interface class which contains the majority of the methods for manipulating wireless interfaces. There are two static functions detailed below provided by the main module. If an error occurs during the execution of a method an exception is raised and the method will return *NULL*.

get_interface_list() Returns a list of the wireless network interfaces on the node.

get_interface(ifname) Returns an instance of the *iwinterface* class that is bound to the interface specified by the *ifname* parameter.

The *iwinterface* class returned by the *get_interface* call provides the following methods.

set_essid(new_essid) Sets the ESSID of the interface to the value specified by the *new_essid* parameter.

get_essid() Returns the current ESSID of the interface.

set_mode(new_mode) Sets the mode of the interface to the value specified by the *new_mode* parameter. This value must be one of the *IW_MODE_** constants listed below.

get_mode() Returns the current mode of the interface.

set_channel(new_channel) Sets the channel of the interface to the value specified by the *new_channel* parameter. The channel may not be specified when the interface is in Managed mode. .

get_channel() Returns the channel the interface is currently configured to.

clear_spy_addr() Remove all MAC addresses from the interfaces spy list.

add_spy_addr(addr) Add the MAC address specified by the *addr* parameter to the interfaces spy list.

get_spy_addr(addr) Returns a dictionary containing signal quality statistics for the link to the MAC address specified by the *addr* parameter.

get_stats() Returns a dictionary containing general statistics regarding the wireless interface.

scan_all() Triggers an 802.11b scan from the wireless interface and returns a list or *iwscanresult* objects representing 802.11b networks that were discovered.

An *iwscanresult* object returned by the *scan_all* function has the following properties.

cellno An index into the scan results, incremented for every network discovered

cell The cell id (BSSID) of the discovered network.

essid The network name (ESSID) of the discovered network.

mode The operating mode of the discovered network.

freq The frequency the discovered network is found on.

chan The channel index that represents the *freq* parameter.

qual A dictionary containing approximate signal quality statistics for the discovered network.

The *iwtools* module makes all of the *IW_** constants defined by the Linux Wireless Tools available to Python programs.

Appendix C
Example Quagga Configuration

ospfd.conf

```
! -*- ospf -*-
!
! CRCnet Wireless Router
!

hostname mwh
password ****
enable password ****

interface mwh-int

interface ful
        ! Link Mode (Ad-Hoc)
        ip ospf cost 50

interface mms
        ! Link Mode (Ad-Hoc)
        ip ospf cost 50

! Router Process
router ospf
        ospf router-id 10.1.27.254

        passive-interface mwh-int
        network 10.0.0.0/8 area 0

! No log file for Biscuit PCs
```

Bibliography

[1] Daniel Aguayo, John Bicket, Sanjit Biswas, Douglas S. J. De Couto, and Robert Morris. MIT roofnet implementation. http://pdos.lcs.mit.edu/roofnet/design/, Accessed October 2003.

[2] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 121–132. ACM Press, 2004.

[3] J. Broch, D. B. Johnson, and D. A. Maltz. The dynamic source routing protocol for mobile ad hoc networks. *IETF Internet draft, draft-ietf-manet-dsr-01.txt*, Dec 1998 (work in progress).

[4] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual International Conference on Mobile Computing and Networking (MobiCom'98)*, October 1998.

[5] LAN MAN Standards Committee. ANSI/IEEE Std 802.11, 1999 edition. Technical report, IEEE Computer Society, 1999.

[6] LAN MAN Standards Committee. ANSI/IEEE Std 802.3, 2002 edition. Technical report, IEEE Computer Society, 2002.

[7] Core Security Technologies - Pcapy. http://oss.coresecurity.com/projects/pcapy.html, Accessed October 2004.

[8] CRCnet. http://www.crc.net.nz/, Accessed October 2003.

[9] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146. ACM Press, 2003.

[10] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers, and Robert Morris. Performance of multihop wireless networks: Shortest path is not enough. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, New Jersey, October 2002. ACM SIGCOMM.

[11] Yih Chun Hu, Adrian Perrig, and David B. Johnson. Efficient security mechanisms for routing protocols. In *Proceedings of the Tenth Annual Network and Distributed System Security Symposium (NDSS 2003)*, pages 57–73, February 2003.

[12] IETF MANET working group charter. http://www.ietf.org/html.charters/manet-charter.html, Accessed October 2004.

[13] Java 2 Runtime Environment 1.4.2 Installation Notes for Linux. http://java.sun.com/j2se/1.4.2/jre/install-linux.html, Accessed October 2004.

[14] Jangeun Jun, Pushkin Peddabachagari, and Mihail L. Sichitiu. Theoretical maximum throughput of IEEE 802.11 and its applications. In *Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications (NCA'03)*, pages 249–256, 2003.

[15] Sumit Khurana, Anurag Kahol, and Anura P. Jayasumana. Effect of hidden terminals on the performance of IEEE 802.11 mac protocol. In *Proceedings of the 23rd. Annual Conference on Local Computer Networks*, pages 12–20, 1998.

[16] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[17] Thanasis Korakis, Gentian Jakllari, and Leandros Tassiulas. A MAC protocol for full exploitation of directional antennas in ad-hoc wireless networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 98–107. ACM Press, 2003.

[18] Alberto Leon-Garcia and Indra Widjaja. *Communication Networks: Fundamental Concepts and Key Architectures*, chapter 7.5, pages 522–534. McGraw-Hill, second edition, 2003.

[19] Locustworld. http://www.locustworld.com/, Accessed October 2004.

[20] Amos Aked Swift (NZ) Ltd. Review of telecommunications infrastructure to provide access to data services in small communities and towns. Report, Ministry of Economic Development, 2001.

[21] Henrik Lundgren, Erik Nordströ, and Christian Tschudin. Coping with communication gray zones in IEEE 802.11b based ad hoc networks. In *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, pages 49–55. ACM Press, 2002.

[22] Joseph P. Macker and M. Scott Corson. Mobile ad hoc networking and the IETF. *SIGMOBILE Mob. Comput. Commun. Rev.*, 2(1):9–14, 1998.

[23] MIT Roofnet. http://www.pdos.lcs.mit.edu/roofnet/, Accessed October 2003.

[24] J. Moy. OSPF version 2. Technical report, IETF, Mar 1994.

[25] NIST kernel AODV implementation. http://w3.antd.nist.gov/wctg/aodv_kernel/, Accessed October 2004.

[26] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector (DSDV) for mobile computers. In *Comp. Commun. Rev.*, pages 223–244, Oct 1994.

[27] C. E. Perkins and E.M. Royer. Ad hoc on demand distance vector (AODV) routing. *IETF Internet draft, draft-ietf-manet-aodv-02.txt*, Nov 1998 (work in progress).

[28] Provicial broadband expansion project. http://www.probe.govt.nz/, Accessed October 2004.

[29] [quagga-dev 1378] Dynamic Reconfiguration of Quagga (OSPF). http://lists.quagga.net/pipermail/quagga-dev/2004-August/001377.html, Accessed October 2004.

[30] Quagga routing suite. http://www.quagga.net/, Accessed October 2004.

[31] Quagga routing suite - pending patches. http://www.quagga.net/pending/, Accessed October 2004.

[32] Ram Ramanathan. On the performance of ad hoc networks with beamforming antennas. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 95–105. ACM Press, 2001.

[33] Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, April 1999.

[34] Seattle Wireless. http://www.seattlewireless.net/, Oct Accessed October 2004.

[35] Soekris Engineering. http://www.soekris.com/, Accessed October 2004.

[36] South Hampton Open Wireless Network. http://www.sown.org.uk/, Accessed October 2004.

[37] W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff. *Unix Network Progamming Volume 1*, chapter 29.5, pages 792–793. Addison Wesley, third edition, 2003.

[38] The Libnet Packet Construction Library. http://www.packetfactory.net/projects/libnet/, Accessed October 2004.

[39] K. Varadhan. Classless inter-domain routing (CIDR): an address assignment and aggregation strategy. Technical report, IETF, Sep 1993.

[40] Wireless Leiden. http://www.wirelessleiden.nl/, Accessed October 2004.

[41] Wireless network device. http://www.wand.net.nz/projectDetail.php?id=95, Accessed October 2004.

[42] Wireless Tools for Linux. http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html, Accessed October 2004.

[43] Su Yi, Yong Pei, and Shivkumar Kalyanaraman. On the capacity improvement of ad hoc wireless networks using directional antennas. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 108–116. ACM Press, 2003.