Abstract

The deployment of wireless wide-area networks presents many challenges. This project describes a system to allow individuals with minimal training to deploy such networks quickly and easily. This project focuses on the development of a robust, easy to use link-establishment routine that assists users in configuring both physical and logical network layers.
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Proposed Future Architecture</td>
<td>35</td>
</tr>
<tr>
<td>5 Project Summary</td>
<td>37</td>
</tr>
<tr>
<td>A Packet Structures</td>
<td>38</td>
</tr>
</tbody>
</table>
List of Figures

1  An example wireless wide-area network. .......................... 2
2  Overlapping of 802.11 channels. ................................. 9
3  Modes of operation in IEEE 802.11 networks. ...................... 10
4  Radiation patterns for omni-directional and directional antenna . 12
5  Required accuracy for $8^\circ$ beam-width and no positioning error. 13
6  Worst case positioning error. ..................................... 13
7  Adjustment of required positioning accuracy ....................... 14
8  A Rapid Deploy Node ................................................ 16
9  The insides of a Rapid Deploy Node. ............................... 17
10 Structure of the mtd software. ..................................... 18
11 Invitation Procedure. ............................................... 19
12 The two loss cases for the Invite Procedure. ....................... 21
13 A functioning Rapid Deploy Network. .............................. 25
14 A Dinsmore 1655 compass sensor. ................................ 28
15 High-level diagram of compass unit ............................... 29
16 Close-up of Compass and Gerbel controller ....................... 30
17 Expected output from compass sensor. ............................ 31
18 Measured output from compass sensor. ............................ 31
19 The complete antenna positioning system ......................... 34
20 Invite Packet Structure. ............................................ 38
21 Accept Packet Structure. ............................................ 38
1 Introduction

Wireless technologies are providing opportunities for research into new and exciting ways of using computer networks. One of the most promising features of wireless technology is that the reach of the network is no longer bound by the length of the cable. Networks can now be deployed with less constraints on their physical topology and so become more useful. With this lack of reliance on a wired medium comes certain complexity. Users wishing to deploy such networks not only need knowledge of networking fundamentals but also need knowledge of principles involved in wireless transmission, such as which antenna to use in a certain situation or how to allocate radio spectrum efficiently to maximise link quality and throughput.

Wireless wide-area networks (WANs) allow us to connect multiple wireless stations, or nodes, together to form a routeable, multi-hop network. Using this architecture, we can create networks that, for example, span many kilometres and may branch out in separate directions. It is these wide-area networks that will be the focus of this study.

Figure 1 shows a three-node wireless wide-area network. Node 1 acts as a gateway to the network by connecting to an external network, for example, another WAN or the Internet. Node 2 acts as a repeater, with two wireless interfaces, one connected to Node 1 and the other to Node 3. Node 3 acts as both a repeater and an Access Point for client devices, such as laptops or PDAs to connect to.

To deploy even a simple wireless network such as the network illustrated by Figure 1 requires a significant amount of knowledge. It is the aim of this project to develop a platform that allows users with little knowledge of wireless technologies and networking to create large scale, high-performance wireless networks quickly and easily. Such a platform could be used in many applications, for example:

- Rapid deployment of wireless networks could be used by agencies such as Civil Defence in order to quickly deploy communication networks into disaster areas to more efficiently manage relief efforts. This would be particularly useful in situations where existing communications infrastructure has been destroyed or disabled by the disaster. To facilitate this, it is a requirement that the wireless network is able to be deployed extremely
quickly, without specialised knowledge. It also requires that the network be robust enough to work in such conditions.

- Other uses could include broadcasting of events from remote locations, for example, streaming of video and audio from a music event. Networks to accommodate this need are usually temporary in nature and must be able to be set up by non-technical users quickly and easily. A rapidly deployable wireless network would be able to provide the infrastructure necessary for streaming video and audio at a fraction of the cost of leasing a fixed line from a telecommunications provider.

- Rural or urban communities could deploy their own wireless networks to enable sharing of Internet access between individuals or to assist community businesses by providing data networks where there would otherwise be none. This use would require the system to be easily deployable by users with little to no knowledge of the underlying technologies.

This project will look at the challenges involved in creating large-scale wireless networks and attempt to create a platform that allows users to deploy wireless networks rapidly, with little knowledge of the underlying technology necessary.
CR Cnet, the wireless research network run by the WAND Network Research Group at the University of Waikato provides an excellent example of a large-scale wireless network. This network has been developed and deployed over the last four years. It services nine rural schools and approximately 12 houses by providing Internet access at speeds much greater than available over existing copper twisted-pair telephone lines. By creating a rapidly deployable wireless networking system, networks such as CR Cnet could be deployed by communities themselves over a much shorter time frame.

1.1 Project Aims

This project aims to help users overcome some of the difficulties in setting up wireless networks, especially those which relate to deployment of large scale networks in a short time period. The following challenges and their proposed solutions have been identified.

1.1.1 Ease of use

The ultimate goal of the project is to create a rapidly deployable wireless networking system. The interface to set up each node in the network must therefore be easy to use for an average person with minimal training. The user interface should be intuitive and easy to use. It should also be able to be used in a wide range of lighting conditions. For example, laptop screens can be very hard to see in direct sunlight. These outdoor conditions should be taken into consideration when designing the rapid deploy system. Removing the need for laptop computers when configuring a wireless node and providing an interface on the node itself is a goal of this project.

1.1.2 Performance

A rapidly deployable wireless network should be able to support many different applications, from simple Web browsing to streaming voice and video content. Thus, the rapid deploy network should configure itself in such a way to sustain high-data rates and provide the infrastructure necessary to allow redundant wireless links if required. Users should be able to expect performance equal to currently available wireless networks.

1.1.3 Frequency Allocation

In order to achieve the goal of high performance, this project will implement a transparent frequency allocation algorithm, removing the need for users to be
aware of the problem of frequency allocation. The system will automatically calculate the best frequency on which to operate a link without any input from the user. The system will have to ensure that links are not configured such that links which are physically close to one another operate on the same or overlapping frequencies, as this will cause radio interference — similar to a poorly tuned television or radio — which decreases data throughput as well as the stability and reliability of the data link.

1.1.4 Antenna Alignment

To increase the range of wireless networks means using higher gain, more directional antenna. As the distance between nodes increases it becomes harder to position directional antenna correctly. Directional antenna (which are explained in Section 2.4) need to be aligned precisely to achieve maximum signal strength. The problem of aligning the antenna becomes much harder when the distance becomes so great that we are unable to physically see the other end of the link. A system is needed for assisting users in orienting antenna precisely, and will attempt to align the antenna accurately to achieve maximum signal strength. The problem of antenna alignment requires highly accurate knowledge of both position and heading. The Global Positioning System (GPS) described in Section 2.5 will be used for calculating position. To determine heading, we will compare several different methods, including using electronic compasses and calculating heading through multiple GPS.

1.1.5 Automatic Network Configuration

As illustrated by Figure 1, setting up even a small wireless wide-area network requires a significant amount of knowledge. The multi-hop nature of wireless wide-area networks causes the complexity of configuring such networks to increase as the number of nodes in the network increases. The user should not need to know about how the network shifts traffic. In particular, the user should not need to worry about the IP networks and host addresses for each link, nor should they need worry about the routing protocols used. A link in the network should be abstracted as much as possible. The user should have no knowledge of Network Address Translation, gateway routes, DNS proxying, etc. These concepts are explained in Section 2.6
1.1.6 Client Devices

Once the network has been deployed, we need a method for client devices such as wireless laptop computers or Personal Digital Assistants (PDAs) to connect to and use the network. Common in homes and offices today are Wireless Access Points. These devices provide wireless network access to an existing (usually wired) network. This same functionality should be available to users of the rapid deploy network. Once the network has been deployed, users should be able to use their laptop or PDA to connect to the network and the external WAN (such as the Internet) that it is connected to.

1.2 Report Structure

This section has given a brief introduction to the need for rapidly deployable wireless networks. Section 2 will discuss related work and provide the background information necessary for an understanding of the topic of rapidly deployable wireless networks. Topics covered include a discussion of existing wireless technologies, some basic antenna theory, GPS, and a brief discussion of routing protocols. Section 3 will cover the design and implementation of the Rapid Deploy network hardware and software system. Section 4 will outline proposed improvements to the Rapid Deploy architecture and a summary of the project is given in Section 5.

1.3 Summary

This project aims to provide a hardware and software platform that allows users to quickly and easily create large wireless wide area networks. These networks will be able to provide access to external networks, such as the Internet, as well as provide a system for end users to access the network through wireless access points. Users will be assisted in setting up the physical links by a system that automatically positions antenna based on GPS location. The interface for setting up such a network will be provided on each node, and will abstract enough of the link-establishment procedure to make it easy for users with little technical knowledge to set up the network.
2 Background

This project relies on utilising a number of existing technologies. The details of these technologies will be discussed further in this section. This section outlines some of the details of the IEEE 802.11 specification, different types of antenna, the Global Positioning System and details about the routing protocols used.

2.1 Related Work

Wireless networking is a hot topic for research at the moment. There are several projects in active development that aim to provide easy to use wireless networks. However, most of these projects focus on a single-antenna mesh structure, with links being formed with neighbours automatically. The most notable of these projects is the MIT Roofnet project[1]. Roofnet relies on a single omni-directional antenna on each node, limiting the network to a single channel. Because of this, Roofnet does not scale up well. For example, if the node density is high, there will be many nodes within radio range of each other, all operating on the same frequency. The IEEE 802.11 protocol ensures that nodes within the same network on the same frequency do not interfere with each other by only allowing a single node to transmit at a time. This severely limits total data throughput. Inversely, if the population of nodes is small, then the network may become partitioned because of the use of relatively low-range omni-directional antenna.

Midkiff et al [9] have developed a rapidly deployable broadband network specifically designed for disaster and emergency response. Their work focuses on the development of link-layer technologies and protocols designed to allow hub stations to blanket areas with wireless coverage. Remote stations then connect back to the hub and provide Ethernet connectivity to the local area. This approach forms networks with “star” topologies, similar to the use of Infrastructure mode in IEEE 802.11. The work is highly focussed on disaster and emergency management, and includes development of applications to help the user place hub stations effectively. Because the project uses custom link-layer protocols, they are restricted to radio technologies that allow customisation of the link-layer. This eliminates most low-cost, consumer-grade wireless equipment. The use hub and remote stations restricts the network to a star topology, and makes deployment of a wide-area, multi-hop wireless network difficult. However, it does provide an easy to use system for deploying wireless networks within a disaster area as well as network management tools.
In 1997, Minden et al [11] proposed an ATM-based rapidly deployable radio network (RDRN). It relies on two types of nodes, Remote Nodes (RNs) and Edge Nodes (ENs). The RNs form an ATM WAN, and the ENs provide access to the WAN, serving as radio access points or base stations which provide connectivity between other users of ENs and the ATM WAN. A unique feature of the RDRN was that it consisted of two overlayed radio networks, a low-bandwidth omnidirectional network for configuration of the network, and the high-bandwidth directional network to connect RNs and ENs together. The RDRN system achieved speeds of up to 10 Mbit/s. The main difference between RDRN and the Rapid Deploy system developed in this project is that RDRN was based on ATM link-layer technology, rather than Ethernet, a more common link-layer today. Again, the RDRN project focussed on developing robust link-layer protocols for automatic deployment of the network. This project focusses on using cheap, off-the-shelf hardware where possible.

Last year, Matt Brown, a COMP420 student, developed a mesh networking system that utilises multiple radio interfaces on each node [2]. The system automatically searches for other nodes in the area and creates a database of possible links. It creates links based on signal-to-noise ratio and other factors. Each node broadcasts its link database so that all nodes have a consistent view of the network. OSPF is then run on top of the network to handle routing. However, the network does not give the user any chance to decide which links to create, and assumes that the node has 360 degree wireless coverage through the use of multiple sectional antenna. There is also a significant protocol overhead as the nodes are constantly exchanging “hello” packets that contain their entire link databases, similar to a distance-vector routing protocol.

Over the summer of 2004–2005, Blake Watkins and I worked on developing a simple prototype rapid deploy network. This involved researching the requirements for such a network and resulted in the development of a software system written in the Python programming language which was able to manage multiple IEEE 802.11b radio interfaces, allowing the user to select from other nodes in the area to connect to. The resulting system was limited to simple network layouts using a single technology. It had preliminary support for allowing access to outside networks. This work will form the starting point for this project.
2.2 Wireless Technologies

We can usually separate wireless technologies into two distinct categories. The first category of wireless technology operates in radio spectrum that requires a licence from the government to use. Use of licenced spectrum is generally only available to large companies, as the cost of both the licence and the technologies required to run at such frequencies is usually prohibitive. An example of such a technology is that which is used for cellphone networks. An advantage of using a licenced wireless system is that the organisation does not need to worry about other users broadcasting on the same frequency causing interference. This leads to higher throughput and better reliability.

The second category encompasses wireless technologies that operate in spectrum covered by a general licence, issued by the government for public use. In New Zealand, as well as many other countries, there is a general licence for operating in the 2.4 GHz and 5.8 GHz bands. As long as users do not exceed certain output power levels, then they are free to use the spectrum as they wish [16]. The downside to this is that the public spectrum is sparse, and so there are often many devices trying to use the same frequencies thus causing interference. For example, microwave ovens, cordless telephones and home wireless networks all operate in the 2.4 GHz band. The IEEE 802.11 standard, also known as “Wi-Fi”, utilises publicly available spectrum to provide wireless networking services. It is an open standard, and as such is widely implemented and well understood. Hardware that implements the IEEE 802.11 standard is relatively cheap compared to hardware used with privately owned spectrum. There are also products that utilise the publicly available spectrum but use proprietary link-layer protocols other than 802.11. One such technology is provided by Trango Broadband Wireless, which operates in the 5.8 GHz band. Proprietary solutions such as Trango Wireless provide much better throughput and reliability over long distance links when compared to IEEE 802.11 devices. An aim of this project is to create a framework so that any type of wireless networking system, be it licenced or unlicensed can be incorporated into the rapid deploy architecture. As a proof of concept, this project will focus on 802.11b technology.

2.3 IEEE 802.11

Today, IEEE 802.11 is the industry standard for setting up wireless local area networks (WLANs). Developed by the Institute of Electrical and Electronics Engineers and approved in 1999, the standard has become ubiquitous with indoor WLANs. IEEE 802.11 was designed to operate in publicly available radio
Figure 2: Overlapping of 802.11 channels.

spectrum to provide a means of wireless networking without the need for expensive licences. The IEEE 802.11 standard defines operation in the 2.4 GHz and 5.8 GHz bands and provides data rates of 1 Mbit/s and 2 Mbit/s[5]. This project will utilise the IEEE 802.11b standard, which operates solely in the 2.4 GHz band, but provides data rates up to 11 Mbit/s while maintaining backwards compatibility with IEEE 802.11. 802.11b was chosen because hardware and support for IEEE 802.11b is widespread and relatively cheap. 802.11b is preferred over 802.11g which provides even higher data rates (up to 54 Mbit/s) because 802.11b is able to operate more reliably over longer distances. There is, however, no reason why 802.11g products could not be incorporated into the Rapid Deploy architecture for use over short links.

2.3.1 802.11 Channels

IEEE 802.11 uses the Instrumentation, Scientific and Medical (ISM) frequency bands (2400–2483 MHz and 5725–5780 MHz) to transfer data via microwaves. The ISM bands are available in most countries to use without a licence, however manufacturing equipment that operates in these frequencies requires a licence. IEEE 802.11b uses the 2.4 GHz range which it splits into 11 channels (see Figure 2). These channels are 22 MHz wide and their center frequencies are spaced 5 MHz apart, which causes some of the channels to partially overlap[4]. Channels allow multiple separate physical networks to run in parallel in the same air space. However, if two networks operate on channels that use a portion of the
same radio spectrum, then they will interfere with each other and thus cause a drop in the aggregate total data-rate. As channel utilisation increases, several factors cause the total data-rate to drop. A large number of frame errors occur as stations cause packets to collide, hence they must be retransmitted. When collisions are detected, some 802.11 devices will decrease the bit-rate at which they are sending. This causes the channel to become unavailable for longer periods of time, slowing down the total data-rate for all stations [7]. Only three of the 11 channels (1, 6 and 11) are completely non-overlapping, that is, they will not interfere with each other, and so efficient allocation of channels is a key consideration in the design of a wireless network.

2.3.2 802.11 Modes of Operation

The IEEE 802.11 standard specifies two main modes of operation for wireless networks (Figure 3). In Ad-Hoc mode, wireless stations talk to other stations on the network directly — there is no single point where data must be routed through. Ad-Hoc mode was designed so that small groups of users could make wireless networks with each other quickly and easily, such as in a meeting or conference, without the need for additional infrastructure.

The second mode is called Infrastructure mode and consists of a base station usually referred to as an Access Point or Master station through which clients, referred to as Managed stations, must connect to before becoming part of the network. Use of Access Points allows the network administrator to put
restrictions on which Managed stations can join the network. Forcing users to connect through a single Access Point also has the advantage that the base station can force parameters such as which channel to use, or what bit-rates are allowed. Also, when using Infrastructure mode, routing is left to the Master station whereas in Ad-Hoc mode routing must be performed dynamically at each node, and so requires extra software to be running to implement multiple hops.

The Rapid Deploy Network uses both Ad-Hoc and Infrastructure mode. Ad-Hoc mode is used for the point-to-point links between nodes, which will usually be provided by long distance, directional antenna. Infrastructure mode is used when an interface is configured to be an Access Point for the network and will usually consist of a single omni-directional antenna or several sectional antenna. Using the directional antenna for the Ad-Hoc links means we can obtain large distances between two nodes. A more thorough discussion of antenna follows.

2.4 Antenna

The subject of antenna design and operation is extremely complex. This section attempts to explain the basics necessary to get an understanding of the problems involved in choice and alignment of antenna. The theoretical “perfect” antenna is known as the isotropic antenna that radiates energy in all directions equally. The radiation pattern of the isotropic antenna can be visualised as a sphere surrounding the antenna. This antenna is a physical impossibility, but it serves as a base from which to compare real antenna. Antenna gain is a measure of the directionality of an antenna compared to the isotropic antenna with antenna loss taken into account[13]. In other words, it tells us how much “better” an antenna is in a certain direction than the isotropic. Gain is usually measured in decibels (dB), a unitless measure which tells us the relative amount of power radiated in a certain direction compared to the theoretical isotropic antenna. Each increase of 3 dB means that the power output in a certain direction is increased by a factor of $2^{[12]}$.

There are two main types of antenna that we will deal with. The first is the omni-directional antenna. These antenna radiate power in all directions in the horizontal plane and typically have gains between 3 dBi and 8 dBi. They are useful for blanketing an area with coverage but because the radio energy must be spread over a large area most omni-directional antenna do not have a very long range. To get radio links over long distances we must use directional
antenna. With directional antenna we can see gains of around 24 dBi as they focus most of the radio energy in a single direction. This focusing causes the effective beam width to become smaller as the gain of the antenna (and hence its useful distance) increases. Hence, with a highly directional antenna we have a relatively small beam width compared to an omni-directional antenna with a lower gain.

This poses problems when trying to get two high-gain directional antenna aligned correctly, as the effective beam width is very small. With the 24 dBi antenna that would typically be used in the Rapid Deploy Network, we have a beam width of approximately 8°. With such a small beam width to deal with, we need an accurate way of orienting the antenna. If we can determine an accurate position (latitude and longitude) of the node we are at, and the position of the node we are trying to connect to then we can calculate the distance to the node, as well as the bearing we need to position the antenna to. One way to achieve this goal is to use the Global Positioning System (GPS).

Figure 4: Approximate radiation patterns for omni-directional and directional antenna. The shaded area shows the effective area of radio power. The black dot indicates the position of the antenna. The “horizontal” row shows a top-down view, whereas the “vertical” row shows a side-on view. The diagrams are not to scale.
2.5 Global Positioning System

The Global Positioning System was developed by the US Department of Defence to replace the ageing TRANSIT system for global navigation. It utilises a constellation of 24 evenly spaced satellites which orbit Earth every 12 hours. This provides the minimum necessary coverage of four satellites in view anywhere on earth at any point in time. Receivers on Earth can listen to the transmissions from the satellites and calculate the distances to the satellites based on the time delay between when data was transmitted and when it was received. This is the basic principle behind the operation of GPS. Using standard GPS, with good satellite geometry we can expect an accuracy of approximately 10 meters[6]. However, in practice perfect satellite geometry is hard to achieve, so the estimated accuracy can fluctuate as satellites come in and go out of view.

![Diagram of satellite beam width and position](image)

Figure 5: Required accuracy for $8^\circ$ beam-width and no positioning error.

Let us assume that we have an effective beam-width of $8^\circ$. We can prove that the accuracy of the GPS system will be sufficient for most uses of the Rapid Deploy network by using simple trigonometry. If we assume a minimum link distance of 1000 meters, then at 1000 meters, we have a beam width of 140 meters. This gives us the diameter of the circle that the remote station must lie in in order to be within the beam. Assuming a perfect case (Figure 5), where

![Diagram of worst-case positioning error](image)

Figure 6: Example of worst-case positioning error. Here, STA 1 is 70 m South of its estimated position, and STA 2 is 70 m North of its estimated position. STA 2 is no longer within STA 1’s beam width.
the remote station lies in the center of the circle, the radius tells us the accuracy required from our positioning system for the bearing calculation to work, which in this case is 70 meters.

However, because we are looking to define a minimum accuracy, we should use the worst case scenario. In this worst cast scenario, both stations have mis-predicted their position by the maximum amount possible. For example, the local station is actually 70 meters North of where it has predicted, and the remote station is 70 meters South of where it has predicted, as in Figure 6. So, in order to account for this, we must make the circles smaller — that is, increase the required accuracy. Halving the diameter of each circle brings both stations back into beam-width given an $8^\circ$ accuracy, so our required accuracy from our positioning system becomes 35 meters, which is well within the limits for GPS (Figure 7).

![Figure 7: The required accuracy of the positioning system has been adjusted to 35 m. STA 2 is now within STA 1’s beam-width again.](image)

### 2.6 Network Layer Routing

Creating the physical links is not enough to create a usable network. Once the physical links are made, higher level services must be configured. The most basic of these services is IP addressing and routing. IP addressing is covered later, as it is handled by the Rapid Deploy Software. Calculation and distribution of routing information is beyond the scope of this project, so we will leave the job to an established routing protocol. However, the Rapid Deploy software still needs to configure the routing daemon that is running on each node as well as assign correct IP networks to each link.

There are many routing protocols available, but they mostly fall into one of two categories. The first general category is distance-vector protocols. These protocols, such as RIP, transmit their entire routing tables to all other incident routers. On reception of a routing table, routers will add routes that they do
not know about to their routing table, and will invalidate stale routes that have expired. Periodically sending routing advertisements can leave the network in an inconsistent state for the period of time between routing advertisements when interface states change between them, and on large networks, the protocol overhead can become quite high.

The second type of routing protocol is the link-state protocol. This protocol type differs from distance-vector protocols in that instead of transmitting the entire routing table periodically, routers will send small link-state advertisements (LSAs) to incident routers only when the state of an interface on the router changes. This greatly decreases the amount of network traffic, and also allows other routers to become aware of the change much quicker than using a distance-vector protocol.

Open Shortest Path First (OSPF) is a link-state routing protocol based on Dijkstra's Shortest Path First algorithm[3]. An implementation of OSPF is available in the Quagga [10] routing suite for Linux. This project makes use of OSPF to calculate and distribute routing information. Nodes that have a connection to an external network distribute default information — that is, if any nodes require access to a network outside of the Rapid Deploy network, then traffic will be routed to the gateway node automatically. Multiple gateway nodes can exist, and OSPF will use the closest node in terms of hop-count.

## 2.7 Summary

The Rapid Deploy Network System uses a variety of existing technologies to accomplish its goal of providing a networking system that is easy to use and maintain. This project will use IEEE 802.11b hardware as a demonstration of the principles behind rapid deployment of wireless networks while ensuring that other technologies can be integrated easily. It will use the Global Positioning System to get accurate fixes on the position of each node in the network to assist in the orientation of directional antenna by calculating distance and bearing. Routing will be handled by the Quagga OSPF routing daemon to ensure that traffic is routed efficiently and that gateway routes are advertised to the rest of the network.
3 Design and Implementation

The previous sections have discussed why a rapidly deployable wireless network would be useful and have introduced some of the problems (and potential solutions) involved in doing so. The rest of this report will discuss the development of the Rapid Deploy Network software and the accompanying hardware that was developed to support it.

Each Rapid Deploy Node (see Figures 8 and 9) is made up of a Soekris 4526 embedded microcomputer housed in a hard plastic case. The Soekris runs a minimal Linux distribution based on Debian “Sarge”. Each node of this type can support up to two wireless interfaces, with larger Soekris computers able to support up to five. Each node has a 20x4 character LCD display, buttons for user input and houses a sealed lead-acid battery that can last for several days. The nodes run the Rapid Deploy software, as described below, in order to allow them to connect to other Rapid Deploy Nodes and form wireless wide-area networks.

Figure 8: A Rapid Deploy Node
3.1 rnetd - The Rapid Deploy Network Daemon

A major portion of the project was to develop control software to run on the Soekris microcomputers to control and manage the wireless network interfaces. The first choice to be made was to decide on a programming language to use. The Python scripting language was chosen over the likes of C, C++ or Java for several reasons. A language was needed for quick prototyping of the system, and Python allows us to develop complex applications quickly, with language support for lists, tuples, dictionaries and other useful data structures. Stable storage space on the Soekris microcomputers is limited to 64 megabytes, and so the size of the runtime environment is a major concern. The runtime environment for Python amounts to only a few megabytes, whereas the Java Virtual Machine alone is larger than the total available space. C and C++ were discounted, as Python allows for more rapid development of applications. This rapid application development comes at a runtime speed cost, however the runtime speed of C or C++ compared to an interpreted Python program is not necessary for this project.

The software that runs on each node is called rnetd, and is a set of Python modules that interact to manage the wireless network interfaces (cards) and the user interface. The structure of the system can be seen in Figure 10. rnetd runs as a background process on startup, and is configured from a simple configuration file that specifies, for example, any default interface states, and the serial device that the user interface is attached to. This configuration file is not designed to be accessed by normal users, but is used when configuring nodes.
before they are shipped.

The *rnetd* module is the main entry point to the system. It starts up *rnetif* objects for each wireless interface on the node, each of which have an associated *antenna* object, which communicates with the motor and compass sensors. The *rnetd* object creates the *rnetui_lcd* object to interact with the user interface hardware on the node. Each *rnetif* object communicates with the Linux kernel wireless extensions system via Python language bindings.

Each node must have a unique identifier between 0 and 255 inclusive in order to operate on the Rapid Deploy Network. The NodeID is used for various functions, such as IP address assignment, which will be discussed later. Each node can have multiple wireless interfaces attached to it which are numbered from 1 to n. When the node is started up, the software initialises the interfaces and places them into a listening state, waiting for invitations from other nodes. Each interfaces ESSID (a string used by the 802.11 protocol to identify networks) is set to “rdr*nodeID-interfaceID”, so interface 2 of node 1 will have an ESSID of “rdr*1-2”. The interface is set to Ad-Hoc mode, so that it can easily create temporary networks with other nodes in order to invite them. The Ad-Hoc interfaces begin broadcasting their presence via the usual 802.11 method of beacon frames, which other nodes can pick up and discover their neighbour.
nodes from.

The user is presented with a simple interface which allows them to select a specific wireless interface on the node and use it to search for other interfaces on other nodes in the area. To do this, an 802.11 scan is performed on the selected interface. Results are filtered in such a way that only 802.11 stations that are in Ad-Hoc mode with an ESSID prefix of “rdn*” are shown. Also, interfaces from the current node are not shown and neither are interfaces that are already part of a link. A user may select any of the available interfaces, at which point the user is presented with a screen that displays signal and noise information, based on beacon frames from the selected remote interface. It is here that the user should position the antenna in order to achieve maximum signal-to-noise ratio (SNR). Once a satisfactory SNR has been achieved, the user can initiate the Invite procedure.

3.1.1 Inviting remote nodes

In order to expand the network, we need to invite new nodes. The interface on the node from which we are issuing the invite is called the local interface. The interface on the node which we are inviting is called the remote interface. Once a user has selected a remote interface to invite, the local interface initiates the Invite Procedure (Figure 11). The local interface switches to the ESSID and

```
Local interface

ESSID (channel)  |  ESSID (channel)
rdn*1−1 (1)       |  rdn*2−2 (1)
rdn*2−2 (1)       |  invite (6,11...)
rdn*1−1*2−2 (6)   |  accept (6)

Remote interface
```

Figure 11: Invitation Procedure.
channel of the remote interface in order to create a Layer 2 network with it, in order to send Ethernet packets. The local interface sends a list of available channels to the remote node, which chooses the best channel from the list and sends an accept packet with the chosen channel. The remote interface switches to the Link ESSID, which is formed by concatenating the local and remote interface ESSIDs, in the form \texttt{rdn*localNodeID-localIFaceID*remoteNodeID-remoteIFaceID} and sets its channel to the chosen link channel. The local interface changes to the Link ESSID and channel upon receipt of the accept packet.

3.1.2 Channel Allocation

Efficient channel allocation is an important part of setting up a wireless network, so an algorithm was needed so that nodes could, in a distributed fashion, select the appropriate channels for links. To achieve this, each node contains a list of available and in-use 802.11 channels. This list is ordered such that preferred channels are at the start of the list. For example, it is preferred on an 802.11 link that channels 1, 6 and 11 be used first, as they do not overlap each other (refer to Section 2.3.1) and hence appear first in the list. The rest of the list is ordered to minimise interference. When an interface sends an invite, it sends the current list of available channels to the remote station in the Invite packet. The remote station then finds the first channel in the list it receives that is also in the list of available channels on its node. The selected channel is then sent back to the inviter in the Accept packet. Assuming that the channel lists are ordered correctly, we can ensure that links will be on the best possible channel for the two nodes. This scheme however does not take into account other wireless networks in the area which could be already using preferred channels. A scan of the 802.11 spectrum before inviting could remove channels that are already in use, but this has not been implemented yet.

3.1.3 IP Addressing

Assigning ESSIDs and channels to a link is not enough to create a routeable network. The system still needs to allocate Internet Protocol (IP) addresses and form routes. IPv4 addresses are four bytes long and are usually denoted by writing each byte in decimal form separated by ‘dots’, such as 192.168.1.10. Every IP address in the Rapid Deploy Network starts with “10.10” as its first two bytes. Each node has a single-byte identifier which should be unique over the network. It is used as the third byte of the address to ensure each node has a unique network prefix from which to allocate addresses. Given this, each
node has 256 addresses from which to allocate to links in the network. In a point-to-point system, each point-to-point link is actually a very small network, consisting of four addresses. The first address is the network address, and is used to describe the whole network - in this case a point-to-point link. The next two addresses are allocated to each end of the link. The fourth address is the broadcast address, an address to which any packets sent are broadcast to all of the addresses in the small network. Given this, with 256 addresses to allocate from it would be possible to allocate 64 point-to-point links, each with their own network address, broadcast address and two host addresses.

For example, if the local node has a Node ID of 2, then the first available network address will be 10.10.2.0, with 10.10.2.1 and 10.10.2.2 being assigned to each end of the link, and 10.10.2.3 being assigned as the broadcast address. The first available network address (in this case 10.10.2.0) is sent to the remote node as part of the Invite packet. The remote station sets the IP address of the invited interface to the upper of the two host addresses (10.10.2.2) and sends the Accept packet. The local station receives the Accept packet and sets the inviting interfaces IP address to the lower of the two host addresses (10.10.2.1). A record of network addresses in use is kept, and is updated as links come up and go down, so that we can check for and avoid conflicts when assigning network addresses.

3.1.4 Error Conditions

Before the Invite Procedure is complete, the only method of communication between two nodes is using the Ethernet protocol, an unreliable transport pro-

![Figure 12: The two loss cases for the Invite Procedure.](image-url)
In order to have reliable communications between nodes, the Internet Protocol must be set up, which is what the Invite Procedure is designed to achieve. However, wireless communication is inherently lossy, and so the Invite Procedure can be subject to packet loss which causes the two nodes to enter an inconsistent state. There are two loss cases, both of which are detected by the local interface as a missing Accept packet after an Invite is sent.

Figure 12 shows the two loss conditions. The first involves the Invite packet being lost in transit. This can be remedied by a simple re-transmission of the Invite packet after a timeout period, as the remote station is still in its initial configuration state. The second case is more sinister. The Invite packet is received and processed by the remote interface, but the Accept packet gets lost in transit. The remote interface, assuming that the Invite Procedure is over, sets itself to the Link ESSID and channel. However, the local interface times out and realises that the Invite Procedure failed. A simple re-transmission of the Invite packet will not be sufficient, as the remote interface has changed ESSID and channel, stopping further communication with the local interface. The local interface has no way of knowing whether it was the Invite or Accept packet that was lost, and so now cannot communicate with the remote interface.

There are two possible solutions to this problem. The first involves a constant "keep-alive" packet being transmitted between interfaces that have links with each other. If the keep-alive packet is not responded to a certain number of times, then the interface can assume that the link is down and reset itself to a known state. For example, the remote interface would start broadcasting keep-alive packets once it has sent an Accept packet, and when several fail to get a response it can reset itself to its normal state, upon which the local interface can re-invite the node.

The second method involves, upon detection of a failed Invite Procedure by the local interface, performing another 802.11 scan and looking to see if any interfaces have a Link ESSID that includes the local interfaces ID when it should not. For example, given the loss of the Accept packet in Figure 12, the local interface could perform an 802.11 scan. It would see a node with the ESSID of "rdn*1-1*2-2". It knows, however, that a link between 1-1 and 2-2 does not exist, and so can presume that the lack of an Accept packet was due to a loss of the Accept packet from the remote interface, rather than the loss of the Invite packet. The local interface can then assume that the Accept packet was supposed to get through, and set the ESSID and channel accordingly. The
second method is preferred, as nodes may be left alone for several days after
being set up. If, for example, a tree falls between two nodes and the nodes
cannot communicate with each other for several minutes they will reset and the
link will be destroyed. The obstruction could then be removed, but the nodes
no longer have a link, leaving the network broken. Users would then have to
visit the nodes and set the link back up again, which could prove difficult in
a disaster situation. The second method ensures that a link gets created and
stays created and hence has been implemented by the Rapid Deploy software.

3.1.5 Client Devices

It is possible for an interface to be put into 802.11 Master Mode and act as a
standard access point for client devices to use to connect to the Rapid Deploy
Network. The user only needs to switch its operating mode from “Point-to-
Point” to “AP” from the user interface to do this. When this is done, any link
involving the current interface will be taken down and the remote station will
be put into normal a normal listening state. The local interface then switches
to Master mode, and reconfigures the DHCP server on the node to include
its address in the list of available DHCP servers. DHCP allows end-users to
automatically connect to a network by letting the network provide all of the
information necessary to the client in order to connect.

At present, we assume that a node has the entire 256 IP addresses from which
to allocate point-to-point networks. This is not actually the case, as we need IPs
to allocate to interfaces acting as Access Points, and also to allocate to clients
connected to the network through the Access Point interfaces via DHCP. So,
the actual IP allocation is as follows. Assume we are looking at node 1. The
address range 10.10.1.0 to 10.10.1.127 are reserved for point-to-point networks.
This gives us a total of 32 point-to-point networks able to be allocated per node.
The range 10.10.1.128 to 10.10.1.131 are reserved as addresses to be allocated
to interfaces going into Access Point mode. This means that we can have up
to four Access Point interfaces per node. The rest if the IPs (10.10.1.132 to
10.10.1.255) are reserved for client devices allocated by DHCP.

A DHCP server runs on each node, and is re-configured to accept DHCP re-
quests from clients on interfaces that are configured to master mode. It hands
out addresses in the 10.10.nodeID.128/25 network from 10.10.nodeID.132 to
10.10.nodeID.254 inclusive. The DHCP server also sets the Domain Name Sys-
tem (DNS) server address to the address of the first Access Point on the node.
This means that any DNS queries originating from a client device will go to the node itself, rather than to a central DNS server. Each node runs a DNS proxy server which is set to talk to the proxy server residing on the node's default gateway address. This way we make a network of proxying DNS servers which all eventually talk to the external DNS server that is connected to the network. This stops us having to distribute external DNS information through the network and has the added benefit of caching DNS lookup results at each node.

3.1.6 Gateway Nodes

Access to the Internet or other WAN can be enabled by setting a node to be a “Gateway Node”. This special mode is activated via a configuration file, but will eventually be able to be activated on any node through the user interface. A Gateway Node connects to a WAN through its wired Ethernet interface. Usually, a Gateway Node will request a DHCP lease on the wired Ethernet interface when it starts up. Once a DHCP lease is acquired, it sets Network Address Translation (NAT) to run over its gateway interface. This makes the entire Rapid Deploy Network look like a single machine to the outside network, reducing the need for exterior routing information to be passed around. The Gateway Node will then reconfigure OSPF to broadcast the fact that it has a gateway route, which can then be distributed to all of the other nodes on the network.

3.1.7 Other Features

The current version of the Rapid Deploy Network Daemon software has several other features that deserve mention. Once a link is active, a user may request a throughput test to be performed on the link. This is done using the Iperf software, which creates a TCP/IP session and attempts to send as many packets as possible through the link in ten seconds. This gives the user an idea of how effective a link is, and is another metric to use as well as SNR when deciding which links to create.

A standard 802.11 scan can be performed if the node has a wireless card that is not in use on a link at the time. This can be used to help get a feel for what other wireless networks are available in the area before setting up a node. At present the channel allocation algorithm does not take other wireless networks in the area into account when deciding which is the best channel to use, but it is planned to do so.
Figure 13: A functioning Rapid Deploy Network. The node on the left is connected to the Internet and to the node on the right. The node on the right has a second radio interface that is acting as an Access Point for the laptop to connect to the network.

The user can also query the node to see the IP address allocations for each of the interfaces. This is especially helpful when diagnosing problems with the network.

3.1.8 Testing and Results

So far the Rapid Deploy Network software has been tested in the lab environment, with three nodes, one of which is acting as a gateway to the CRCnet research network. The second node acts as a repeater and the third node acts as a final repeater and access point to the network. The results of this test network have been very pleasing. We are able to create a three node network with a wireless laptop connecting and surfing the Internet in several minutes. Each of the links on the network operate at approximately 4 Mbit/s, which is fairly good when using 802.11b hardware, which has a theoretical maximum throughput of 6.1 Mbit/s[8]. During testing of the network, it has become apparent that different wireless chipsets give quite different results in terms of throughput. For example, Orinoco based chipsets seem to operate at approximately 3 Mbit/s, whereas Prism based chipsets operate at around 4 Mbit/s. This is likely a driver issue, but is more a curiosity than a concern.
3.2 Antenna alignment

With the software able to create links and form routeable multi-hop networks, we now require a means for helping users to align antenna to maximise signal strength. The problem of antenna alignment stems from the relatively small beam-width of highly directional antenna. To create links of significant distance, we need to use these directional antenna, which typically have a beam-width of $8^\circ$ or lower. When the remote node we are connecting to is many kilometres away, it can be very difficult to manually align the antenna at each end. Another factor motivating the automation of antenna alignment is safety. Antenna are usually mounted on poles or in high places. The less time the user needs to spend in such a position the better, especially as manual alignment of antenna is a very delicate and time-consuming task.

In order to align antenna correctly, we need a knowledge of the current position of the two nodes being connected together in terms of latitude and longitude. With this information we can easily calculate a bearing to position the antenna to. However, if we are to automate the process, we also need to know the current heading of the antenna, so that the system can determine how far it needs to move the antenna. The system implemented in the Rapid Deploy system is described below.

3.2.1 Position

As explained in Section 2.5, the Rapid Deploy system uses GPS as a way of determining the position of nodes. GPS is able to give us a fixed position estimate to within a certain accuracy, given knowledge of the satellite geometry. This position estimate is based on estimating the distance to each satellite in view and determining an area that the receiver could be located. The signals received from the GPS satellites can be affected by radio-frequency noise and as such can affect the resulting position estimate. When a GPS receiver is placed in a fixed position the resulting position estimates are usually close to normally distributed in both the latitude and longitude axes. This is caused by noise as well as clock synchronisation errors. In order to try and overcome the error inherent in the GPS system, moving averages of the latitude and longitude fixes can be taken. As well as this, several different types of GPS were looked at in order to find a good cost to accuracy compromise.

Three different methods of using GPS have been investigated. The first method was to use an expensive off-the-shelf multi-channel GPS receiver. The
receiver that was tested was a Garmin eTrex Summit. The unit cost approximately NZD $600 and gives us an accuracy down to about 5 meters given good satellite geometry and a clear view of the sky. This accuracy is achieved by applying special filtering to results, given satellite geometry information. The filtering algorithms are considered proprietary by Garmin, and as such are not available to the general public. The Garmin unit also outputs its current heading using an internal electronic compass. It outputs all of the information through a serial port running at 4800 baud, using the NMEA 0813 format. The amount of data the Garmin outputs each update is enough to cause each update to take almost 2 seconds at 4800 baud, so the information we receive could be up to 2 seconds old. The speed of the serial port cannot be increased on the Garmin unit, as this would break NMEA compliance.

The second method was to use a less expensive GPS unit that does not have all of the features of the Garmin. The GPS tested was a Sony TP-051 serial port based GPS receiver. This model does not have the built in compass however it also outputs data at 4800 baud in NMEA format. The Sony unit tracks less satellites at once and does not apply special filtering of results to get a better estimate of its position. The unit costs about NZD $80, and does not output how accurate it thinks its fixes are. Python modules were written that can listen to NMEA data coming in from either the serial port or a file, and analyse it. At present, the application stores relevant fixes and performs moving averaging of latitude and longitude. The accuracy of this method is hard to quantify without performing prolonged tests at an accurately known location. For now, however, we have to take it for granted that the GPS locks are “accurate enough”. This is a definite area of improvement that could be investigated, but for the purpose of finishing a proof-of-concept, it has been left at that.

The third and final method tried was to use several of the cheaper GPS units at the same time, and triangulate a fix from the data from the multiple GPS units. This method is much more complex, requiring that each of the GPS units be positioned an exact and known distance away from the other units. We would need to take an estimate the position of each GPS, analyse the error in each of the fixes and calculate from there the possible actual positions. The amount of work needed to bring such a system to fruition would be more than it is worth. From preliminary results it seems that there are too many uncontrollable variables such as noise and bias involved in the GPS system for such an accurate measurement to be taken.
This project uses the second method, and in trials so far the GPS receivers with moving averages do seem to be “accurate enough”. The Rapid Deploy software was modified so that a GPS can be connected to the node, which is continuously sampled. The samples are averaged and the result can be obtained by querying the user interface.

3.2.2 Heading

Once the software knows the latitude and longitude of the node, it can calculate bearings to other nodes. However, in order to automate the process, the software also needs to know the current heading of the antenna in order to calculate how far it needs to move it to bring it to the correct bearing. There are two parts to this equation, ascertaining the current heading, and moving the antenna to a specific heading.

To ascertain a compass bearing we can either use motors that have a system for indexing its current position and then align the motor with a known heading, or use an electronic compass on the antenna to get a real-time reading of its heading. Another option is to use a gyroscopic system, which gives us tilt and roll information as well, which would be useful for aligning antenna that are at significantly different altitudes. However, gyroscopic based systems are extremely expensive and are not practical for this project. Indexed motors are cheaper than gyroscopes, however are still too expensive, and still require the user to align it to a known heading first. An electronic compass is much cheaper and allows us to know the current absolute heading of the antenna whenever we need it.

This project uses a Dinsmore 1655 electronic analogue compass sensor (Figure 14), which takes a 5 volt input voltage and has two outputs. As the compass sensor is rotated, the two outputs resemble a sine and cosine wave. The theoretical
output is shown in Figure 17. An initial experiment was conducted to measure the actual voltage output of the sensors. This was achieved by connecting two volt-meters to the outputs and recording the results as the compass sensor was rotated. The results are shown in Figure 18. An arbitrary point on the sensor was chosen as a reference to zero, which explains why the output waves appear to be out of phase slightly from the theoretical output. The relative position of these two waves represents a heading, which we must interpret.

Getting the compass sensor to a usable state was not a trivial task. Figure 16 shows the final set-up required. This consists of a “Gerbel” micro-controller board running control software written in C and compiled using the Small Devices C Compiler. The control software takes analogue inputs through two of the Gerbel's Input/Output ports, sends them to the Gerbel ADC converter and then prints the resulting digital representation to the serial line in hexadecimal. Two output lines are dedicated to signalling the motor controller connected to the Gerbel. The first output line represents the state of the motor (on/off), and the second represents the direction (clockwise/anti-clockwise). The motor controller contains two relays that switch the motor to the correct state based on the outputs from the Gerbel. Figure 15 shows a high-level view of the Gerbel system. Various problems arose from electromagnetic interference due to current switching which would reset the Gerbel, causing its internal state to become inconsistent with the Rapid Deploy software on the host node. The problem was fixed by having the Gerbel output its current state to the serial port each time it changes so that the Rapid Deploy software can detect inconsistencies. Shielding all sources of electromagnetic interference, including wrapping the motor

Figure 15: High-level diagram of compass unit
assembly in tin-foil helped to reduce the number of Gerbel resets that occurred. This also helped the compass sensor to become more accurate, as it was not being influenced by local magnetic fields.

### 3.2.3 Compass algorithm

An algorithm was needed to convert the incoming waves to actual compass bearings. An algorithm was devised and implemented as a separate Python module so that it could be useful in other applications. In order to describe the algorithm at work, we will use Figure 17 as a reference. Notice that at approximately 3.3V and 1.7V the two waves intercept. Between these two voltages, the waves are relatively straight lines and there is only ever one wave between the two crossing points (the middle section). From this, we can devise a piecewise function of x, the bearing. The “pieces” of this function are made up of sections in the graph where one wave is in the middle section of the graph, and the other is either above the upper crossing point or below the lower crossing. For each section we can uniquely identify it by the relative positions of inputs A and B. For example, we are in section 1 when input B is above the “upper crossing line” ($y = 3.3$ in this case), and input A is in the middle section. By mapping appropriate trigonometric functions to these sections we can derive a
Figure 17: Expected output from compass sensor.

Figure 18: Measured output from compass sensor.
useful bearing. The piecewise function is described below

\[
\theta = \begin{cases} 
\arcsin n_a & y_b > c_{\text{upper}}, y_a > m_a \\
-\arcsin n_a + \pi & y_b < c_{\text{lower}} \\
\arccos n_b & y_a > c_{\text{upper}} \\
-\arccos n_b + 2\pi & y_a < c_{\text{lower}} \\
\arcsin n_a + 2\pi & y_b > c_{\text{upper}}, y_a < m_a
\end{cases}
\]

Where \(y_a\) and \(y_b\) are the two inputs from the compass and \(n_a\) and \(n_b\) are the respective normalised values. \(c_{\text{upper}}\) and \(c_{\text{lower}}\) are the upper and lower crossing points respectively and \(m_a\) is the mid-point of the A input wave.

The actual version of the algorithm implemented in software goes on to take into account cases where the points are within a certain threshold of the crossing points, to dampen out the effect caused by the algorithm switching between cases when the input data is jittery. Because there is no zero reference on the compass sensor itself, the output will need to be adjusted by some constant offset, which can be found by orienting the sensor to a known bearing. The difference between the known bearing and the calculated output should be used as the constant offset, which is subtracted from each reading thereafter.

The compass algorithm relies on knowing the upper and lower crossing points, as well as the mid points of each wave. In order to calculate these values we need to know the amplitudes of each wave. The amplitudes of the waves can vary slightly in different electromagnetic environments — enough to cause the algorithm to break down if used with old calibration data. To remedy this a calibration mode was implemented in the Python compass module, which continuously reads in data while the compass is being turned through a full rotation. We can then calculate the amplitudes from the minimum and maximum values, and then calculate the y-values of the two crossing points, \(c_{\text{upper}}\) and \(c_{\text{lower}}\), by applying the following formula.

\[
c_{\text{upper}} = a_a \sin x + m_a \\
c_{\text{lower}} = a_a \sin (x + \pi) + m_a
\]

where \(x = \arcsin \frac{(m_a - m_b)}{\sqrt{(-a_a)^2 + a_b^2}} - \varphi\)

and \(\varphi = \begin{cases} 
\arctan \frac{n_a}{-a_a} & -a_a > 0 \\
(\arctan \frac{n_a}{-a_a} + \pi) & -a_a < 0
\end{cases}\)
Where $c_{upper}$ and $c_{lower}$ are the y-values where the two waves cross. $m_a$ and $m_b$ are the mid-points of the A and B input waves respectively and $a_a$ and $a_b$ are the amplitudes of the A and B input waves. This formula was derived from the formula given for detecting phase shifts in [14]. A non-zero amplitude is assumed.

Using the above formula we can calibrate the compass sensor modules to adapt to changing magnetic environments. The final feature of the compass module is the ability to save the current calibration data, which can be reloaded at a later date if the node is in the same place. If, for example, a node runs out of power, a new battery can be connected and the node can begin to function immediately without the need for re-calibration.

3.2.4 Results

Given that nodes now have a knowledge of their location and the bearing of their antenna, there is one final step in automating the antenna alignment process. When a user selects a target node, they can also enter the GPS location of that node — presumably communicated over CB radio between users — and the local node can calculate the bearing to the remote node by applying the following formula which is based on spherical geometry:

\[
D_{rad} = 2 \cdot r_{earth} \cdot \arcsin \sqrt{\frac{\sin^2 lat_1 - lat_2}{2} + \cos lat_1 \cos lat_2 \sin^2\frac{long_1 - long_2}{2}}
\]

\[
C_{rad} = \arccos \frac{\sin lat_2 - (\sin lat_1 \cos D_{rad})}{\cos lat_1 \sin D_{rad}}
\]

Where $D_{rad}$ is the Great Circle Distance between the two points, and $C_{rad}$ is the bearing from $(lat_1, long_1)$ to $(lat_2, long_2)$. $r_{earth}$ is an approximation of the radius of Earth at the location we are measuring. The value for $r_{earth}$ is problematic, as there are several approximations to choose from. 6367 km is widely regarded as a good approximation, based on the definition of the nautical mile. It lies between the currently accepted WGS84 equatorial and polar radii of 6378 km and 6356 km respectively[15].

The node then signals the Gerbel micro-controller to switch the motor to the necessary state and samples the compass while the motor is engaged. When the bearing of the antenna comes within a pre-defined threshold (currently one degree), the node can signal the Gerbel to switch off the motor, or adjust the position if it has gone too far.
Figure 19: The complete antenna positioning system

Testing of the antenna alignment procedure has been very pleasing. We have been able to position antenna to the location of remote nodes to within several degrees. This is well within our $8^\circ$ beam-width. More testing is needed over longer ranges before we can be sure that the system works well in all situations. The accuracy of the system degrades significantly over short distances, partly due to inaccuracies in the GPS locks, as well as general rounding error in the formulae used. However, it is not envisioned that this system will be used unless links are significantly long, that is, over several kilometres.
4 Proposed Future Architecture

This project has demonstrated the viability of a rapidly deployable wireless networking system and has investigated and solved many of the challenges involved in developing such a network architecture. However, there is plenty of room for improvement and this section outlines a possible evolution of the current state of the rapid deploy architecture.

The current rapid deploy architecture allows many different types of wireless technology to be integrated into the system, however the implementation of different technologies requires certain "hacks" in order to abstract the link details to the user. For example, consider the integration of a point-to-point wireless system that operates in pairs, where each pair must be specifically programmed with the unique ID of its partner before it will communicate with it. In this case, two random Rapid Deploy nodes would not be able to communicate with each other over this point-to-point link unless the radios had been programmed manually. This would require manual configuration of the radios as well as careful planning of the topology of the network before deployment, two issues that the Rapid Deploy network proposed to remove. Ideally, we would want any of the point-to-point radios to be able to communicate with any other point-to-point radio in the field, whether it had been pre-programmed or not. This would require communication to already be available between nodes before the link is established, in order to configure each of the radios with the unique identifier of its new partner.

The Rapid Deploy software assumes that users are aware of the GPS coordinates of the nodes when they are attempting to establish links. Nodes already know their own GPS locations, which are communicated to the users via the user interface. However, the users still need to communicate their location to the other nodes through some other channel, for example CB radio. Ideally, the nodes could communicate their locations to each other without the need for the user to become involved. However, this cannot be done until a radio link is established. This simple detail stops the network from being completely automated. It would be preferred if the user could simply choose a node to connect to and have the nodes communicate their locations before setting up the main radio link.

The problem of simplifying the integration of new technologies and the problem of communicating position could be solved by introducing a technology
independent communications layer, to be used to boot-strap the main network. This could be achieved by including a low-frequency radio network that connects geographically distant nodes with each other without the need for directional antenna. The low-frequency nature of the radio link would not allow for high data rates but it would be sufficient to boot-strap the higher-powered radio equipment that operates at higher frequencies. This way, each node can be aware of its GPS location and broadcast both its location and the available network hardware it has to other nodes. Nodes can then calculate possible links based on their available hardware and present the user with a list of links to choose from.

In the case of the point-to-point system, the nodes could use the management network to configure pairs of radios automatically, by each communicating their unique identifiers over the low-frequency network before configuring the nodes. The node can then automatically position the antenna without the need for the user to know the GPS location of the remote node, as it has already been discovered through the management network.

A prototype low-frequency radio system has been in development over the course of this year at the WAND Network Research Group. In its current state it can only be used in a point-to-point fashion, which is unsuitable for use as a management layer for the Rapid Deploy network. However, it is able to span large distances without the use of a directional antenna and could be modified to run in a multipoint-to-multipoint mode. Integrating such a system would be the first step in developing a next-generation Rapid Deploy architecture.
5 Project Summary

This project has demonstrated the viability of a rapidly deployable wireless networking platform. The project has succeeded in achieving the main goals of providing a consistent, easy to use interface for deploying such a network and providing a system for automated positioning of antenna. A set of Python modules have been developed to manage the wireless network interfaces on a Rapid Deploy node, which provides the user interface through which the node is configured. A system for automated positioning of antenna has been devised, which brings together several separate technologies. A single, low-cost GPS receiver is used to determine position and a custom built electronic compass sensor module and micro-controller provide heading information. The information from both these systems are processed by Python modules on the host node, and algorithms have been devised to make use of this information.

Testing of the Rapid Deploy system has proved successful. We have been able to create small networks in a lab environment and longer-distance networks in an outdoor environment. In both these cases, the network was deployed with a minimum of user input and the antenna were positioned correctly by the control software. Simulations of even longer distance links have proven that the antenna alignment system works to within the required accuracy.

A future architecture for the Rapid Deploy system has been proposed, which allows the nodes to communicate with each other before the main wireless links are set up. This involves the use of a low-frequency radio network which allows nodes to communicate their position and available hardware information to each other. This allows easier integration of new wireless technologies and removes the need for users to input the GPS location of the node they are connecting to.
A Packet Structures

![Invite Packet Structure](image1)

![Accept Packet Structure](image2)
References


