Partition Anticipation for Wireless Local Area Networks

Andrew Butterly

A dissertation submitted to the University of Dublin,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

September 16th 2002

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: ____________________
Andrew Butterly

Date: September 16th, 2002

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: ____________________
Andrew Butterly

Date: September 16th, 2002

Acknowledgements

I would like to thank my supervisor, Dr. Vinny Cahill, for his guidance throughout the year. Many thanks also go to Mr. Gregor Gärtner, Mr. Raymond Cunningham, and Mr. Fergal O'Hart for their generous gifts of time and advice.

Abstract

Wireless networks introduce new possibilities for computer networking. They also bring some new problems, while enhancing some already established ones. One new problem is a network's increased sensitivity to its environment, while the issues of failure detection, and efficient network routing, are examples of traditional problems that have been exacerbated.

The possibility and severity of partition on a network is another problem that has been made worse by the use of wireless solutions. In the case of wireless partitions, there is less scope for predicting a failure than in traditional networks, or judging it correctly as such once it has happened. This is due to the current lack of experience in predicting the characteristics of the wireless medium, as well its inherent unpredictability.

This document describes and evaluates solutions for a network location information dissemination protocol, and a framework for a wireless network partition anticipator that uses this location information service. The location service presents a process for relative and absolute user location tracking, network view construction, and routing table construction. The partition anticipator framework analyses the output of the location dissemination service and makes inferences for network quality based on an expert system knowledge base. Potential partitions between nodes, as well as excellent quality connections are highlighted as being interesting.

Table of contents

Table of figures

Chapter 1

Introduction

1.1 Motivation

Wireless networks, in particular wireless ad-hoc ones, have an added unpredictability due to the possibility of the movement of the devices involved. For a mobile network to be more effective in roaming environments, it would be useful to be able to make more assertions about that network – specifically in terms of quality and reliability. The ability to infer future network quality, especially possible partitions and node availability, would help to compensate for the unpredictability of a wireless network. One example of this would be the ability to assert that a wireless connection will soon not be viable before it becomes so.

This project is concerned with the development and evaluation of a *partition anticipator* and *network information dissemination service* for wireless networks.

1.2 Wireless Networks

The introduction and success of wireless networks has been driven by the need to lower the installation and maintenance associated with wired network infrastructures, as well as a desire to support networks with mobile facilities. Wireless networks offer benefits in terms of mobility, ease of installation, and ease of use [22].

1.2.1 Characteristics of wireless networks

Wireless networks can be defined as networks made up of communication capable devices that do not have the encumbrance of the wired infrastructure associated with traditional networks. Wireless networks are frequently mobility capable, although this is not a prerequisite. Wireless networks suffer from problems with the quality of their network connections. This is due to the uncertainty of having quality communication in a medium that is not as tightly predictable, or controllable, as a traditional wired network. The mobility factor can often exacerbate this, as node movement will often result in changes in network quality between nodes [19]. In the area of mobile use, wireless communication can be characterised by hard to predict bandwidth and signal loss, and by higher error rates than in wired networks [20].

1.2.2 Definition of a mobile network

A *wireless network* can be defined as a network with nodes not geographically fixed at a single physical location. Wireless networks can exist in an infrastructure mode, where mobility capable nodes can roam relative to fixed access points through which their access is controlled and their traffic is routed. Once a node finds itself moving out of range of a cellular access point, it either finds another willing access point, or finds itself partitioned from the network once out of range of its existing one. This model of use assumes a fixed underlying structure in place that will facilitate access concerns for a node. In this dissertation, situations where this is not the case are considered interesting - where nodes communicate directly to each other, or through each other, without reference to the access point usage model.

For the purposes of this project, a *mobile network* is defined as one made up of mobile capable nodes that may or may not have a predefined notion of the network that they are part of prior to joining it. An ad-hoc network can be defined as a subset of this, where nodes might join or leave a network with no forewarning or reference to any previous agreement between nodes. One major distinction that can be made between mobile networks and ad-hoc mobile networks is the notion of a predetermined structure, that an ad-hoc network is precisely that – ad-hoc and not depending on any predetermined arrangement of nodes.

This work is interested in both mobile and ad-hoc mobile networks, as they have been defined. In this dissertation, the term "mobile network" is used to encompass both predetermined and dynamic mobile networks. A working assumption for this project is that a mobile network is in a constant state of change, as nodes move about a geographical area.

1.2.3 Wireless environments – the notion of a wireless partition

As mentioned previously, the quality of wireless networks can suffer due to the unpredictable nature of the environments that they are deployed in. The characteristics of these environments will be discussed in chapter 3, which examines the chosen wireless technology. For now, it will suffice to say that problems can exist with wireless networks that can lead to a network becoming untenable.

A *partition* is defined as a condition where communication between nodes is not possible. Partitions can arise between wired nodes if the infrastructure connecting them becomes unusable due to any of a selection of possible factors: traffic congestion leading to network throughput being reduced to zero, and a severing of a physical link, to name two. Partitions can be of a temporary nature (for example in the congestion case, it is conceivable that a period of heavy congestion would end, leaving network communication to continue as normal), or have a more lasting effect (for example, a physical break in a wired network that could not be corrected until the link was fixed by a technician).

Partitions can occur in wireless networks due to most of the same reasons that they can occur in wired ones. Thanks to the wireless medium, a physical split in a network cable is no longer a concern, but a connection between two nodes can still be broken if the nodes move beyond each other's range, or if interference blocks the transmissions between them.

At this time it is useful to distinguish between a *logical* and a *physical* partition. A physical partition is one where communication is not possible due to the physical layout of the communicating devices, and a logical partition is one where communication may still be possible, but for safety reasons (i.e. a high probability of an imminent physical partition) the devices involved have chosen not to communicate. In the case of a logical partition, communication would be made impossible at the application layer though some algorithm, but perhaps not at lower layers. This distinction is useful as it allows for the notion of an ‘unsafe region’, where communication is possible but ill advised.

1.3 Concept of a network information service

One interesting service concept that wireless networks have allowed for is that of services that are location specific, or location dependant. An example being a service that has knowledge of its environment, and is capable of using this information in a context sensitive way - e.g. a security badge that causes doors to be opened automatically for its wearer.

One necessity for these services is the need for facilities for location information. In order for a service to be location enabled, there has to first be a mechanism for location determination.

This projects notion of a *partition anticipator* is that of a service that takes in information describing a network in order to make decisions regarding the tenability of that network. In order for this to work, a partition anticipator has the need for a network information service that can provide useful input to it.

Such a service could hold information regarding the quality of the network, the nodes on the network, and even some location information about the nodes connected to the wireless network. An important feature of this sort of information is that it not be stale. To this end, network information statistics should have a notion of their age.

1.4 Concept of a partition anticipator

A partition anticipator is a concept used in [8, 7] to describe a process for receiving information about a set of network conditions, and using this information in order to anticipate periods of interesting network quality. As a rule of thumb, ‘interesting’ can be defined here as conditions that will lead to either good network quality periods, or bad. In the case of [8], a partition anticipator’s observations were used to manage a group membership service. A partition anticipator could be used to anticipate communication failures between mobile devices before they happen.

A partition anticipator (PA) takes as input current information about the network in order to make decisions. A source for such information is a network information service that provides network statistics, such as the one described above. A PA should be able to analyse network statistics and infer possible future states of the network connection – based on some heuristics for network quality.

In this context, a partition anticipator is a useful tool for checking network conditions, the analysis of which can then be used to infer future reliability.

1.5 Objectives for this project

This section lists the objectives for this project, and the achievements that were made.

1.5.1 Objectives for a network information service

Have a basic knowledge of its current position (possibly only relative to other nodes).

Be able to record meaningful statistics for the current network.

To provide a PA service with fresh network information.

1.5.2 Objectives for a partition anticipator

Anticipate / react to node and communication failure.

Have an estimate of a node's coverage on the network.

Make inferences about a nodes future network quality using data about the environment and a knowledge base.

Allow for providing this information to other processes.

Make possible the splitting and merging of groups, or some other higher-level service that a working PA could assist.

1.5.3 Objectives for this project

The analysis of a chosen wireless technology in order to understand its characteristics in different environments.

The discovery of a set of metrics that can be used to judge the quality of a wireless network.

The specification, implementation and evaluation of a partition anticipator for mobile networks. This partition anticipator will use the understanding of wireless networks developed to analyse the network and provide expert knowledge.

1.5.4 Achievements in this project

A comprehensive set of tests on the chosen wireless technology, which were used to define key characteristics of both the technology, and the equipment used to enable the technology.

The development of a network information and location service for mobile networks.

The development of a prototype partition anticipator, plus a framework for modelling expert knowledge to be used by the PA.

1.6 Document Structure

This section outlines the chapters of the dissertation and their contents.

Chapter 1: Introduction describes the motivation behind this project, and introduces the concepts and objectives of the project.

Chapter 2: State of the Art looks at other work that has had an influence on the development of the partition anticipator.

Chapter 3: 802.11 introduces our chosen wireless technology, 802.11b, and describes the tests and analysis undertaken in order to provide metrics and heuristics for out partition anticipator

Chapter 4: Analysis and design looks at the requirements for a partition anticipator and a network information protocol. The issues involved and decisions made in the design of the two are explored and discussed.

Chapter 5: Implementation looks at the implementation of the partition anticipator and location information dissemination protocol. There is an outline of the key algorithms behind the services, and descriptions of interesting properties of the implementation.

Chapter 6: Evaluation presents the constructed partition anticipator / location information dissemination protocol and analyses their merit compared to the aims of the project. This section also discusses the future development and use of a partition anticipator.

Chapter 7: Conclusions concludes the project, and evaluates the success according to the goals defined.

Chapter 2

State of the Art

2.1 Motivation

The following sections will examine papers that use concepts interesting to partition anticipation in four areas: Group communication, routing, location determination, and signal quality prediction. Several of those projects have their own notion of a partition anticipator.

This examination will be followed with a discussion of the original paper that was the basis for this partition anticipator.

2.2 Group communication

2.2.1 Overview

Mobile networking defines an interesting solution for environments where nodes can move through physical space and communicate based on their current location and its characteristics. Group communication, as in wired networks, is a useful paradigm for this: a node can hold a view of the group that it is currently in, and present this view to a process that is running on it as the current makeup of the network. As the network changes, this view will also change. This change in view will allow a process to make decisions, as it will be using data that is at least reasonably fresh. Communication is usually assumed to be symmetric, and mobile hosts will discover each other's presence, and establish communications dynamically [1].

However, the nature of mobile networks does not lend itself to the stability one might require for a group membership system [7] (i.e. one with consistent group views across nodes, reliable and timely delivery of messages, etc.). Timing is often a factor in these networks: the structure of a network can change quickly and group views can become stale / inconsistent with little or no warning.

Mobile groups tend to be transient in nature. Members will drop in and out based on their current preference and availability to a group view [11]. Groups are generally not limited in size (save where the underlying medium forces it). Also, for groups with size limitations/a known set of possible members, generally only a subset of all known members will be part of a group view at a given time. A group may not wish to contain all of the nodes available to it.

2.2.2 Safe distance

Roman et al [1] introduces the concept of *safe distance*. This concept is based around the notion that given two mobile hosts with equal range *R*, the distance between them is deemed *safe* if it does not exceed the maximum distance at which one can guarantee that any communication task taking at most *t* time units can be completed successfully, even in the worst case event of them moving in opposing directions at maximum speed. A group is deemed safe if any two members of the group are connected via a path along which all consecutive hosts are at a safe distance. The notion is also extended to several groups. For example, in a situation with two groups, we require that at least two nodes, one in each group, be at a safe distance.

Safe distance, as defined by [1], does not take into account interferences from external sources, or that the nodes involved may have different receiving signal strengths. It also assumes a known upper bound on node movement speed. By using an absolute positioning system (GPS) to predict future positions, safe distance allows for a system that can propose future breaks in transmission capabilities as well as guaranteeing certain capabilities for a known time at least. A key feature of the algorithm is the use of location information. The leader of a group will frequently check a group member's location in order to make sure that they still qualify to remain in that group.

2.2.3 Proximity Layer

Prakash et al [2] describes an architecture in which a *proximity layer* is inserted between the group membership layer and the underlying mobile network. In this model, a node can belong to several groups.

The proximity layer protocol monitors changes in network configuration using the medium access control (MAC) sub layer primitives. Each node can send a beacon every *t* time units. Such a beacon has a limited range and serves to identify a node, even in the absence of any application level message passing.

Periodically, a proximity test is run on a node that determines if its neighbours are within range - if they are less than a threshold distance away. This information is periodically presented to the group communication layer. At the group communication layer, group membership is built by a protocol that runs upon application demand.

2.3 Routing Protocols

2.3.1 Overview

Routing protocols can be categorised using three descriptors:

- Is it centralized or distributed?
- Is it adaptive or static?
- Is it reactive or proactive or hybrid?

Centralised protocols are those where all routing decisions are made at one node, whereas in distributed solutions all nodes share the routing decisions. Adaptive protocols are those that can change their behaviour based on the current state of the network. The third descriptor is its nature on the network: proactive, reactive or hybrid? Reactive protocols are those that calculate new routes on demand, or in response to the failure of an existing route. They are suited to environments where network links are less likely to fail, such as wired networks. Proactive protocols attempt to maintain routes to all destinations at all times, if they are needed or not. Hybrid systems are a combination of the two.

Proactive protocols have a need for periodic rediscovery of the network and usually a recalculation of the routes therein. This project is interested in distributed, adaptive and proactive routing protocols, as these actively learn about their network, and also adapt to changes in the network while remaining scalable. Some examples of different proactive routing protocols will be introduced now, before some proactive routing projects that have some interesting properties are described.

2.3.2 Example 1: a proactive routing protocol: OSPF

Open Shortest Path First (OSPF) is an adaptive, decentralised, dynamic protocol. It adjusts to problems in the network and only needs a short period to stabilise its routing table. It also prevents routing loops – where a calculated route will go through the same node more than once. It is a link state algorithm, meaning that each node is aware of its immediate links to its neighbours, and that this information is disseminated throughout the network, resulting in every node having sufficient knowledge to build a complete network map.

Periodically, each OSPF router will send ‘hello’ messages out to its neighbours. The neighbours do not record the message, but if a message is not received within an expected time frame, then that node is marked as being down by the OSPF routers. A link state advertisement describing a nodes view of its neighbours and the cost to reach them is generated, and sent out on the network. This way, nodes get advertisements that describe the structure of the network and the cost to one node from another. These advertisements are used to update routing tables held on each node. The OSPF algorithm is inefficient in that it floods the network with link state advertisements, even in the case that no change has happened since the last update.

2.3.3 Example 2: a proactive routing protocol: The Wireless Routing Protocol (WRP)

The WRP algorithm was developed to make more efficient use of bandwidth than traditional proactive routing protocols. Each node holds tables to describe routing, distance, links costs, and message retransmission to the other nodes in the network. WRP is similar to OSPF, with some added complexity to optimise efficiency in a wireless environment. For example, WRP routing updates are only sent to neighbours when something new is discovered, to update information. These updates are driven by changes in topology, and are incremental, only including information that has changed since the last update. The algorithm implements consistency checks of all of its neighbours every time that it detects a change in link on any of its neighbours. It also eliminates looping and has fast convergence.

2.3.4 Notion of a pre-emptive region, signal strength as a threshold factor

Goff et al [3] defines a conceptually similar notion to safe distance: that of a *pre-emptive region*, where a warning is generated when the signal power of a received packet drops below a set "pre-emptive threshold". The paper uses this concept for route table calculation. The warning generated leads to the recalculation of that routing path through the nodes on the network, and a recalculation of the pre-emptive threshold. The idea here is to prevent a path break from happening should a node be moving out of range.

One disadvantage with this technique is that a higher number of paths may be calculated than is necessary where paths become suspect, but do not afterwards break. The new, unnecessary path may be of lesser quality, and the creation of a new path will have increased the pre-emptive threshold such that the algorithm is more cautious – its threshold is bigger, so its effective range is less than before.

In [3], the pre-emptive threshold is calculated using the signal strength of received packets, and the number of hops as a secondary measure. The paper lists two other possible metrics as: the age of a path, or the rate of collisions on it. The mobile network's environment plays a large role here. The paper presumes an open environment where there is a strong reflection of transmitted signals.

2.3.5 Location aided routing (LAR)

This section describes a proactive routing protocol that uses the physical location of mobile nodes in order to discover available routes [9]. LAR attempts to limit the search for new routes to a smaller "request zone" of an ad-hoc network, rather than flooding the entire network as normal proactive protocols do.

In order to determine positioning in an ad-hoc network, the GPS system is used, together with some margin for error on the GPS co-ordinates. LAR models the idea of an *expected zone*, and a *request zone* as subparts of the larger geographical area in which a mobile ad-hoc network is located. An expected zone is the maximum physical range from a node, A, that another node, B, can have travelled within a time period, assuming movement in any direction at a maximum speed. Using this as an upper bound on packet flooding, node A will define a request zone using one of two available schemes. The request zone is the geographical area that node A's ROUTE REQUEST messages for B cannot leave. If another node receives a route discovery request, it will only forward it in the case that it is inside the request zone. In the case of another node not having had previous contact with node B, the discovery request reverts to normal network flooding.

LAR is an example of a 'smart flooding' solution. From the point of view of this dissertation, it is interesting as it allows for absolute positioning of nodes in a mobile ad-hoc network, as well as the storing of information on the nodes of a network about the other nodes. It also provides a solution for intelligent polling for the network in order to discover other nodes.

2.3.6 Comments

Proactive routing protocols discover their network in order to minimise the effects of frequent network topology changes. Tables of routes are kept and updated depending on the protocol used. One key issue is the limiting of the flooding messages that seem to be necessary for discovery.

2.4 Location determination

2.4.1. Overview

User tracking and location detection is an area that has received some attention in recent years. In this section two examples of research that are interesting to a partition anticipator and location information service will be examined. In an outdoor environment, GPS is a useful and scalable solution to the user-tracking problem. In the indoor environment however, GPS is not suitable[1], and so projects have taken to other means in order to track users in the indoor environment. With these indoor and non-GPS efforts, metrics used for user positioning both relative to other users and to fixed access points are worth analysis. Also, user tracking itself is an interesting possibility of use for the network information service introduced in chapter one.

2.4.2: RADAR

The RADAR project [4] is concerned with user location and tracking in an indoor environment using 802.11b nodes operating in access point Mode. GPS is not used to determine position due to its inability to perform indoors. Wireless access points are used as fixed points from which to triangulate statistics from.

Two approaches to user location are considered, both using triangulation techniques and RF technologies. The first is a control technique - empirical data collected from tests around their environment (a floor in the Microsoft building), is used to estimate the position of a node as the smallest Euclidean distance from one of the test points in the area. These tests points have been recorded as distances (in terms of signal strength[2]) from three base stations on the floor mapping to a known location in the building. The second technique sought to replace the cost of discovery of the

[1] GPS gets its co-ordinates by triangulating from a number of satellites. In order for this to work, a GPS receiver must have a good line of sight to at least 3 satellites. Indoors, the multi-path effect results in GPS signals being ineffective due to echoes of received signals being propagated. GPS is not an effective solution indoors [4].

[2] One discovery made in the paper is that signal strength (SS) is a stronger function of location than the signal to noise ratio (SNR). RADAR considers SNR to be impacted by random noise fluctuations, where SS is not. SS was the only factor used to determine a distance metric in the RADAR project. Chapter three defines SS and SNR.

patterns for the environment by the estimation of them according to a suitable RF propagation model.

The environment RF patterns were estimated using this propagation model, while factoring in figures for walls and other possible sources of attenuation. A curious result recorded during these tests was that the orientation of the mobile user had as much as a 5 dBm effect on the received signal strength. This was often the result of a direct line of sight/non-line of sight factor. The high water content in the testers bodies was cited as a factor that affected the tests which would not be easily predictable in a working environment. While there were other sources of fluctuation, such as the movement of people and objects, these tended towards randomness, while the factor of the mobile user was a systematic source of error. To this end, the tests recorded data in each location from several orientations (North, South, etc.).

The second technique proved almost as effective as the empirical one, and the propagation model predictions were very close to the actual results, once walls/other attenuation factors had been taken into account. Results for the empirical technique had a median of 2.94 meters accurate, while the propagation estimations had one of 4.3m accurate.

The results from this project showed that using RF signal strength is enough to determine reasonable accuracy of location in the case of fixed located triangulation points, and that increasing accuracy is possible as the network density increases. Also, the tests run highlighted some unsuspected aspects of radio frequency communication that have important implications for this partition anticipator project.

2.4.3 SpotON: ad-hoc sensor networking

Hightower et al [17] proposes a model of location determination based on both relative and absolute placement. The project proposes the placement of small hardware devices on mobile and non-mobile hosts in an indoor environment. The devices communicate in the 916.5 MHz radio frequencies.

These devices can be configured to know their position in absolute terms (the fixed devices can have GPS co-ordinates set in their memory), or can just infer their position relative to each other.

Devices periodically send each other beacons. These beacons can contain history information to allow nodes to aggregate global snapshots. Received signal strength recorded from these beacons is used to determine the distance between transmitter and receiver. Triangulation is used to infer location.

2.4.4 Comment

An interesting detail from the projects observed here was its use of signal strength as a metric for calculating distance. Another is the rejection of GPS tracking due to the condition of indoor use. RADAR shows the need for tests of a wireless technology with its assertion that a user can affect the quality of a wireless link by standing too close to the network interface. SpotOn stresses the importance of calibration of their radio devices in order to attune them against possible sources of error.

These two projects highlight the question as to what in a wireless technology could be affected by the environment or users movements. Metrics and success factors for partition anticipation will have to take the peculiarities of a chosen technology into account.

2.5 Signal modelling in real time

2.5.1 Overview

The project discussed in [6] is noteworthy as its research goals are similar to those of this project. It also uses concepts of location awareness and partition anticipation, although it is envisaged as having a different use for them.

2.4.2 RF modeling

Punnoose et al [6] deals with real time radio frequency (RF) propagation modelling in multi-hop ad-hoc mobile networking. This project conceptualises a real-time RF propagation model generated independently by each node in a network. The concept of this project is that if changes in a network topology can be predicted, then the job of a routing protocol will become easier as it can re-route traffic in a pre-emptive fashion, before an existing link is even broken. It uses GPS, or any other absolute positioning system to track mobile nodes.

The project designed and implemented a real time predictive propagation model that used terrain information, GPS based positioning for each node, movement modelling / prediction, and propagation modelling to estimate signal strength and loss factor between nodes in a network. As an input, the system takes:

- Absolute positioning information
- Terrain model
- Signal quality input (if available)

This project uses a devised signal strength prediction model in order to predict future network qualities. The positioning information together with the signal quality input is fed into the terrain map and a predicted link quality between nodes is output. Prediction is done on each node individually. Position / motion information about a node is propagated to other nodes by being piggybacked into outgoing messages or sent explicitly.

Aspects to this project that are noteworthy include:

- Facility to affect change on a hardware device using the protocol developed – e.g. change outgoing transmission power, etc
- The use of past and current information on node positions and velocities to predict future physical positions
- The prediction of future network transmission quality between nodes.
- The facilities for real time prediction – that the protocol can be time bounded and have a notion of time-based information.
- The propagation of information gathered on a node to other nodes to assist with their predictions.
- The use of three-dimensional maps in the case where height information is available.

2.4.3 Comment

The project uses detailed terrain maps and the GPS system in order to make its predictions about future positioning and quality of links from nodes. This approach requires accurate absolute positioning, and a previously compiled terrain map that has facilities for mapping co-ordinates of nodes to locations on that map. The project also proposes a non-position based prediction technique that works based on RF patterns alone, although at this time that has not been implemented.

The project is extremely relevant as its design and goals are similar to those of this dissertation.

2.5 The Terminodes Project

2.5.1 Overview

The Terminodes project [11] is concerned with designing a mobile ad-hoc network for public environments. The main focus of the project is that a mobile ad-hoc network should be self-organising. To this end, the project has defined protocols for "the building blocks" of a network. These are solutions for routing, mobility management, GPS / GPS-free positioning, incentives to co-operation, and security. The first three of these will be discussed in light of this dissertation.

2.5.2 Routing in Terminodes

Devices in a terminodes network have a unique identifier and a temporary location dependant address.

Routing is done using a combination of two methods: local and remote routing. Remote routing refers to the case of forwarding messages, which are not close to the destination at present, while local refers to the case where the message is near its intended destination. The remote routing solution uses location information relative to fixed positions in the area to forward messages, while the local routing solution does not. This is on the basis that location information is likely to be accurate in the general case, but less so when the destination draws nearer, in the light of mobile nodes. For the local routing case, a bounded table is kept for neighbours using a proactive route discovery mechanism.

2.5.3 Mobility Management

Terminodes break an area full of nodes into "location reference groups". These groups have a defined centre (the densest node in the group – i.e. the one with the most neighbours). Information about a group is propagated around the network. Groups change physical location as the central node moves around an area.

2.5.4 Positioning

Terminode positioning has the option to use GPS, or its own protocol where GPS is not an option. This protocol is termed a "self-positioning algorithm" (SPA). SPA uses distance measurements between nodes to build a network coordinate system. The time of arrival algorithm[3] is used to obtain the distance between two terminodes. A terminode, *n*, will choose two other local terminodes where they are not in a straight line from the point of view of *n*. The distances between the nodes will be calculated. These figures can be used at a node to calculate the location of a fourth node by triangulation to the other two and itself.

2.5.5 Comment

Terminodes touches on some interesting aspects. Their notions of relative and absolute positioning, used in tandem, are a good solution for preventing over-reliance on a GPS-oriented solution. However, the use of time of arrival as a measure of distance brings extra requirements in terms of timing guarantees. Also, terminodes does not address such real world problems as signal interference, etc.

The "location reference groups" notion of terminodes is a good implementation of a mobile group structure as it allows for local control of a group by the node that is the most popular to all of the others.

[3] The Time of Arrival technique works by measuring the time a signal takes to transmit between two nodes and using this transit time as a metric. In order to remove possible sources of error, time of arrival has strong requirements in terms of timing - for example, the length of time a message spends in an input or output buffer has to be accounted for. Time of arrival is described in [12].

2.6 Comment on the preceding projects / papers

An immediate pattern in all of the mobile enabled protocols that have been looked at is the need for perpetual communication. For example, if a group communication protocol relied on piggybacking group information into messages, then a quiet period would lead to a nodes expulsion from a group. To take a game analogy, it is unacceptable that a player who hides during a game should find himself no longer playing (although it might make for better game play).

Another similarity with these is the concept of threshold: given a certain value of a particular metric, how can a decision-making algorithm decide what is and what is not dangerous in terms of imminent disconnection? We can see from [2] that simply increasing the sensitivity of a threshold value could eventually lead to an untenable network. A threshold algorithm would need to be adaptive to its environment, and have the ability to scale back a threshold in times of more reliable connections.

One non-functional aim for a partition anticipator system such as the one described in this dissertation would be that it is modular enough to use different metrics depending on a given situation. These could be signal strength, path life, GPS co-ordinates, etc. It is important that a failure estimation model to check against be adaptive enough to catch failures and yet not prompt unnecessary ones. Where failure cannot be hidden in a failure detector (for example, a communication failure could not be masked from a higher level protocol), the consequence of that failure needs to be well defined, so that some action might be taken.

2.7 Proximity Groups

2.7.1 Overview

With the group communication paradigm in mind, Killijian et al [7] has proposed two complementary protocols for implementing group-based management with partition avoidance across a mobile network. In that paper, these two feature as unique layers on a protocol stack. These are coverage awareness and partition anticipation. The work of this paper proved to be a starting point for this work in building a partition anticipator and associated information service. The concepts discussed in that paper and their suitability for the needs of this project shall now be discussed.

2.7.2 Mobile groups

Mobile groups are described in [7] as being made up of nodes that are in the effective range of each other. That paper has suggested defining a group based on geographical area: admission is based on a node being located in the geographical area, being interested in the group, and the group not already being full. There the term proximity group is used to describe these sort of mobile groups. Coverage awareness from underlying layers is used to model groups and partition anticipation is used to anticipate possible failures / partitions in the group view.

2.7.3 Coverage Awareness

The next concept is coverage awareness. In this algorithm, a node is aware of the quality of its connections to the network using some metric – e.g. signal strength on a wireless connection to other nodes, or the number of other nodes a device is in contact with. It uses this to build a map of its neighbour nodes and also to determine their location in absolute terms, or relative to itself, at a recorded time. This information is propagated to neighbouring nodes in beacon advertisements sent periodically. Absolute proximity (proximity to a known fixed resource, or the use of GPS co-ordinates to describe a node), and relative proximity (proximity to a non-fixed resource, e.g. another moving object) are conceptualised.

A coverage estimation algorithm can then run. Its functions are to:

- Combine data from available nodes and correlate them into an estimation graphing the area covered.
- Determine (using some internal rule) if an area is being covered and also if there might be areas *not* being covered.
- Set a bound on the propagation of information about a node, to prevent flooding of beacon information beyond a point where that information has probably become stale
- Make the information gathered available to a higher-level protocol on that mobile node (e.g. to a partition anticipator or group communication layer protocol).

2.7.4 Partition Anticipation

The other concept proposed is a partition / failure anticipation algorithm, where a node can receive data from a coverage aware layer underneath it and anticipate nascent failures that may be due to network partition / device failure. Partition anticipation here is the process of predicting possible failures in a group view and responding to them, or recommending a response to a higher layer (e.g. a group communication layer).

In [7], PA is calculated based on a probabilistic approach. The probability of a partition is calculated as being above or below a given *threshold*, and this is used to decide to run in an optimistic or pessimistic mode respectively. That paper uses a combination of *failure anticipators*, *movement planners*, and *environment evaluators* to calculate this threshold. Each monitors for a different failure type. These will be looked at briefly here, as they are interesting in terms of the failure models that they introduce.

2.7.5 Failure Anticipators

These are responsible for monitoring a node, or its neighbours for possible failure. On a local node, ‘failure’ encompasses losing battery power, entering a power saving mode, or crashing in a recoverable way. [7] considers failure models of fail-silent, fail stop, or can recover. A failure anticipator will also look at its neighbours for behaviour that might be an indication of failure: no beacon, decreasing signal strength (when combined with movement planner metrics), etc.

[7] does not discuss Byzantine failure – where neighbouring nodes return inconsistent results about each other, although it is conceivable that a quorum system on each node is intended in order to reach consensus.

2.7.6 Movement Planners

These are responsible for determining the probability of a node failing due to movement. They use the notion of safe distance to calculate this probability within a time frame *t* – perhaps corresponding to the current traffic passing through the network or another rule. If a node is not within a safe distance (determined using location awareness information), it is considered to have failed, even if communication is still possible with it.

2.7.7 Environment Calculators

These can be used to gather and distribute knowledge about environmental conditions over time that may have some shared impact on group and node communications. An example used in [7] is that of a large object moving through the group view in one direction, temporarily disturbing communications in its immediate wake.

2.7.8 Comments

The partition anticipation system outlined in [7] assumes an accurate underlying layer that has coverage aware properties. These two protocols are documented separately, but the partition anticipation layer is heavily dependant on the coverage awareness layer, or another like it providing the same functionality.

The concepts of a partition anticipator and a location information service that are presented in [7] are the starting point for the solutions developed as a part of this dissertation. Where this work has differed from the ideas presented in [7] is due to discoveries made about the nature of the chosen wireless technologies, and in a redefining of the requirements for a PA system. The next two chapters will discuss the chosen wireless technology and show how it has affected design.

2.8 Chapter conclusion

Several of the projects discussed in this chapter propose metrics for various judgements of what constitutes a quality network, as defined by those projects. One metric that has been used consistently across several wireless based projects is that of received signal strength as a good indicator of general quality, and in the outdoor case, a correlation to physical distance. The next chapter will examine if this is true for our wireless technology (the 802.11 standard), as well as attempting to define other characteristics of 802.11 that will help to predict it better.

The other common ground among many of the projects presented is the need for an information service that can provide information to a partition / failure anticipator service. Generally, the information services presented are conceptually similar to the proactive routing protocol network discovery model. Some of the information models also have the added feature of location tracking and location information dissemination.

Chapter 3

IEEE 802.11

3.1 Introduction

IEEE 802.11 is a standard for wireless radio frequency networks. This chapter introduces the 802.11 standard, and a series of experiments carried out on 802.11b networks using two different brands of PCMCIA 802.11b network interfaces. These experiments were carried out by Gregor Gärtner, a member of the distributed systems group (DSG), Trinity College Dublin, and also by the author of this dissertation. The purpose of the tests was to determine which characteristics of 802.11 were interesting in terms of the concept of *partition anticipation*, as it was described in the previous chapter.

3.2 Common terms used in wireless communication

This section will serve to define some of the common terms used in this chapter.

- dBm
    - Decibels relative to one milliwatt. This scale is used to define signal strength for radio and audio frequencies. dBm is based on the decibel,

a logarithmic measure of relative power, and is usually a value in a minus scale.

- Signal strength
  - This is a measure, in dBm, of the power of a signal sent from transmitter to receiver.
- Noise
  - This is a measure of the received noise at an interface, expressed in dBm. Noise can be defined as the radiation received at a wireless interface in the absence of incoming transmissions. Noise interferes with communication by making transmissions indistinguishable from background interference.
- Signal to noise ratio (SNR)
  - This is the ratio of the overall received signal strength to the received noise at an interface. It is expressed in dBm.

3.3 Overview of 802.11

IEEE 802.11 specifies a 2.4 GHz operating frequency [23]. It provides specifications for MAC (medium access control) and PHY (physical) layer functionality for wireless connectivity.

3.3.1 MAC layer

The MAC layer provides access control functions for a shared medium physical layer, in support of a logical link layer (LLC) that is above it. These control functions include addressing control, access coordination, and checksum generating.

Radio transmitters do not permit sending and receiving on the same channel, and as a result of this the 802.11 is designed with avoiding collisions, not with detecting them [19]. The 802.11 standard uses CSMA/CD (carrier sense multiple access with collision avoidance), as opposed to Ethernet, a wired 802 standard that uses CSMA/CD (carrier sense multiple access with collision detection).

3.3.2 Physical layer

The original 802.11 standard [19] defined three physical layer standards. These were frequency hopping spread spectrum (FHSS), direct sequence spread spectrum (DSSS), and infrared (IR). Since then, two others have been defined – high rate direct sequence spread spectrum (HR-DSSS) and orthogonal frequency division multiplexing (OFDM).

The HR-DSSS standard is commonly referred to as 802.11b. It is the most common wireless LAN implementation in use at present. It introduces new features to the 802.11 standard, and allows for data rates of 5.5 Mbps and 11 Mbps. The 802.11b standard is used by this project.

3.3.3 Comments

802.11b operates in an *infrastructure* mode and in an *ad-hoc* mode.

The infrastructure, or access point model is characterised by a base station topology similar to that of GSM, with roaming nodes moving, but staying within range of fixed access points through which all network access is brokered. Access points are chosen using a threshold SNR value, below which a roaming node will consider its current access point untenable [24].

The ad-hoc model is characterised by roaming nodes that are not dependant on access points for network connectivity. Note that, as defined here, *ad-hoc* corresponds to the definition of a mobile network given in chapter one, and hence is the usage model that will be assumed for this project.

3.3.4 Characteristics of wireless communication

The infrared physical layer standard notwithstanding, 802.11 is a radio frequency (RF) based technology. 802.11 RF communication is subject to factors that can affect its quality of transmission. These factors will be discussed here.

3.3.4.1 Path loss

Path loss is the effect of the loss of signal strength between transmitter and receiver. This value is also referred to as the *attenuation* factor. There are several models of path loss, which depend on the environment in which RF transmission is taking place.

3.3.4.2 Multipath propagation

Multipath is the condition whereby a signal sent from a transmitter arrives at a destination from several different signal propagation paths, in addition to the normal direct line of sight propagation. It is caused by three basic phenomena [21]. These will be described now.

- Reflection
    - This is the condition whereby electromagnetic waves are 'bounced' off an object that is large relative to the size of the wave. It results in the propagating wave losing power, and re-propagating in a different direction.
- Diffraction
    - This is the effect whereby an electromagnetic wave is obstructed by an object with sharp edges. The effect on the wave is to make it travel around corners and other edges.
- Scattering
    - This is the effect that occurs when an electromagnetic wave impacts with an object small relative to its size. The wave is dissipated.

The intersection of multipath signals can cause a signal to be increased or decreased. Where multipath waves are out of phase with each other, an echoing effect can occur, which can interfere with the signal received at a device.

3.4 Experiments

3.4.1 Overview of experiments

The experiments carried out on the 802.11b technology were designed to gain a better understanding of the behaviour of an 802.11b network. A suite of tests was designed, and carried out in a large open space. Following this, some basic tests from the first environment were carried out in a different setting, in order to compare quality models for both environments. Both locations were large open spaces with no RF interference at the same or similar frequencies[4], and no line of sight breakage, unless a test required it.

Results from the experiments were analysed using MatLab, an analysis tool. It was also used to generate the graphs shown.

3.4.2 Goals for experiments

The experiments were concerned with understanding network behaviour in 802.11b, and evaluating how a PA could learn about the characteristics of the network in such a way that it could recognise patterns and possibly infer future behaviour. Specifically, the experiments were broken down into seven categories:

- Signal strength on different grounds, and with different brands of wireless interface.
- Height of sender and receiver on different grounds.
- Correlation of frame loss to message length and load.
- Correlation of frame loss to payload pattern.
- Orientation of devices and effect this had on throughput.
- Influence of movement on signal strength.
- Influence of blocking on signal strength.

4 Both locations had GSM coverage, but this was discounted as cellular telephones operate in a different frequency band to 802.11b.

The experiments performed resulted in a large amount of generated data for analysis. It would not be possible to present all of the experiments performed, and the conclusions drawn, without this chapter dwarfing the rest of the dissertation.

As a result of this, this chapter will place an emphasis on the first category of tests run, as these had the most impact on the rest of the project. The rest of the experiments will be described only where they were directly relevant to this version of the location information dissemination protocol and partition anticipator.

This chapter will describe the following categories:

- Signal strength on different grounds, and with different brands of wireless interface.
- One test from the 'Influence of blocking on signal strength' category.

The following will be briefly mentioned in terms of that their results implied, but those results will not be discussed in any major way:

- Correlation of frame loss to message length.
- Correlation of frame loss to payload pattern.

3.4.3 Procedure – experimental set up

3.4.3.1 Overview

The main bulk of the experiments were carried out on Bull Island beach, a nature preserve in Dublin Bay. The beach provided a large flat open space with no obstacles to transmissions. Markers were set up from a chosen point to a distance 360 meters away in a straight line. A marker was placed every 5 meters.

The other environment used was a similar open space in Phoenix Park, Dublin. The same distance markers were installed, and a smaller set of tests completed. This second set was a repeat of the basic tests from the first environment, taken in order to compare the effects of two different surface types.

Where not otherwise stated, during a test one laptop would remain stationary while the other would be carried 360s meters away, and then back again. Communication between the two would break at some point, and resume again on the return route. The laptop computers were held at an approximate height of one meter from ground level, with the laptop screen facing the tester holding it and the wireless interfaces in a PCMCIA slot on the side of the laptops.

GPS devices were used to record movement of the test machines. However, due to problems with these devices[5], the moving tester also incremented a distance value in the program used for testing every time a distance marker was passed. The distance value was decremented on the return route.

3.4.3.2 Testing equipment

The mobile devices used were two Dell Latitude laptops with the RedHat 7.3 operating system. The PCMCIA 802.11b interfaces used were two Orinoco “silver” 11Mbps cards, and in another set of tests for comparison, two Compaq “WL 100” 802.11b interfaces.

A custom piece of software was designed for testing purposes. Its requirements and design are documented in [25]. Its purpose was to send and receive messages between nodes, and record statistics on messages received. All wireless interfaces were set in 2 Mbps mode[6].

Each message would contain a unique sequence number. Sending and receiving of messages between nodes was timed, and nodes would send during assigned windows, during which no other nodes would send. This was to remove the chance of interference from several nodes sending at the same time.

The time on each node was synchronised using the network time protocol[7] before each test was started, and the start up sequence of the software included a message synchronisation procedure to ensure correct timed sending. Using the custom software, one node was designated the ‘leader’, and another the ‘client’. These titles served to identify which node was the one that moved during a test. There was no difference in functionality between the nodes.

[5] The GPS devices used were the Magellan GPS 310. There were problems with the devices in that they were not consistent in the co-ordinates that they returned. For example, when the devices were held next to each other, they reported large distances between them (often between 10 and 70 meters).

[6] The 802.11b wireless interfaces used have a facility for setting the maximum data rate to be used. The lower the data rate, the higher the range that the interface has [26].

[7] Network time protocol (NTP) is a client-server protocol used to synchronise time between a designated server and several clients.

When a device received a message from another device, it would record the event along with values such as the sequence number of the received message, and statistics about the quality of the transmission. These statistics were taken from the wireless interface via the iwspy[8] tool.

3.4.4 Description of experiments

The experiments carried out were broken up into the seven categories listed above. As mentioned above, this chapter will only be able to describe one of these categories, and some parts of others. These will be described now.

3.4.4.1 Category one: Signal strength received on different grounds, and with different brands of wireless interface

This set of experiments was concerned with the following questions:

- What is the range of the different wireless interfaces?
- How does signal strength change with physical distance?
- When are enough packets lost to consider a connection broken?
- Does ground type affect quality?

3.4.4.1.1 Experimental set up

In order to determine the difference in range between wireless interfaces where communication was possible, a normal distance test was carried out on both the Orinoco and Compaq wireless interfaces. This test was carried out three times each to ensure that results were consistent.

Following these tests, the interface with the larger range value was chosen, and used for all of the remaining experiments.

To compare the two identified environments, tests were carried out on the sand and grass settings, in order to determine if one environment had a different effect on multipath propagation than the other.

[8] iwspy is a tool for monitoring network connections on a wireless network. It is part of the wireless extensions package [15].

The rest of this section will detail some of the findings of the tests performed in this category. The results from this section form the basis for the partition anticipator and information service.

3.4.5.1.2 Experiments and results

3.4.5.1.2.1 Range of the difference interfaces

A graph showing average results from the three experiments carried out to compare the range of the two brands of wireless interface is shown below. Each graph shows the distance between the test laptops increasing and then decreasing during a test (on the Y-Axis). The distance values follow a 'step' pattern, as the distance value used was the distance inputted by the moving laptop, which was incremented / decremented by a tester every 5 meters. The X-axis values map the sequence numbers of received messages. Where the lines graphed are broken, no messages were received.

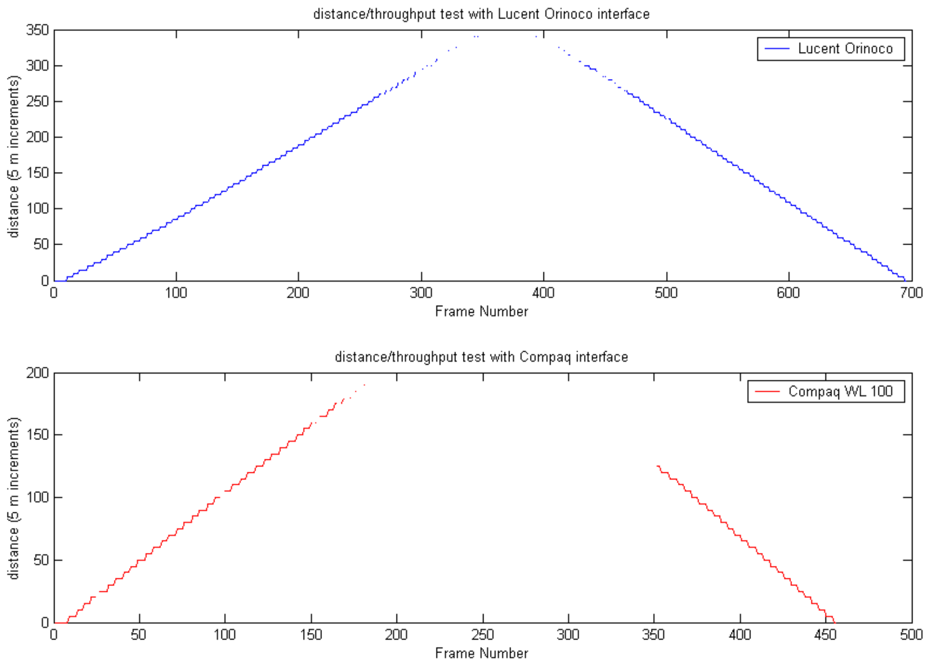


Figure 1, Lucent and Compaq distance to throughput comparison

The lines in the graph show the increase and decrease of distance in the two tests. The Lucent Orinoco wireless interface is able to transmit until the nodes are roughly 290 meters apart, and on the return is able to communicate again roughly around the same value. The Compaq interface is able to communicate until the nodes are 180 meters apart for the first half of the test, then, surprisingly, in the second half of the test, regular communication does not begin again until around the 140 meters distance mark. For both interfaces, after those threshold distances the occasional message might get through, but regular communication is impossible until the distance starts to decrease again. This graph is from the first of three iterations of these particular tests, but the results are roughly consistent across the three iterations done.

Comments

What is remarkable about these figures is that both interfaces were transmitting at the same power level (15 dBm) for the tests, and that the specification both of considers this value to be valid [26,27], yet the Orinoco wireless card outperforms the Compaq by over 100 meters consistently. The same driver was used for both.

This result shows that different network interfaces are quite different in terms of range ability. The rest of the experiments used the Orinoco wireless interface only.

3.4.5.1.2.2 Decrease of signal strength with distance

The following graph shows the decrease in signal strength with an increase in physical distance. It shows the received signal strength in the top picture, and the distance inputted in the lower one. Sequence numbers have been used on the X-Axis in both pictures to show comparison. As before, broken lines indicate message loss.

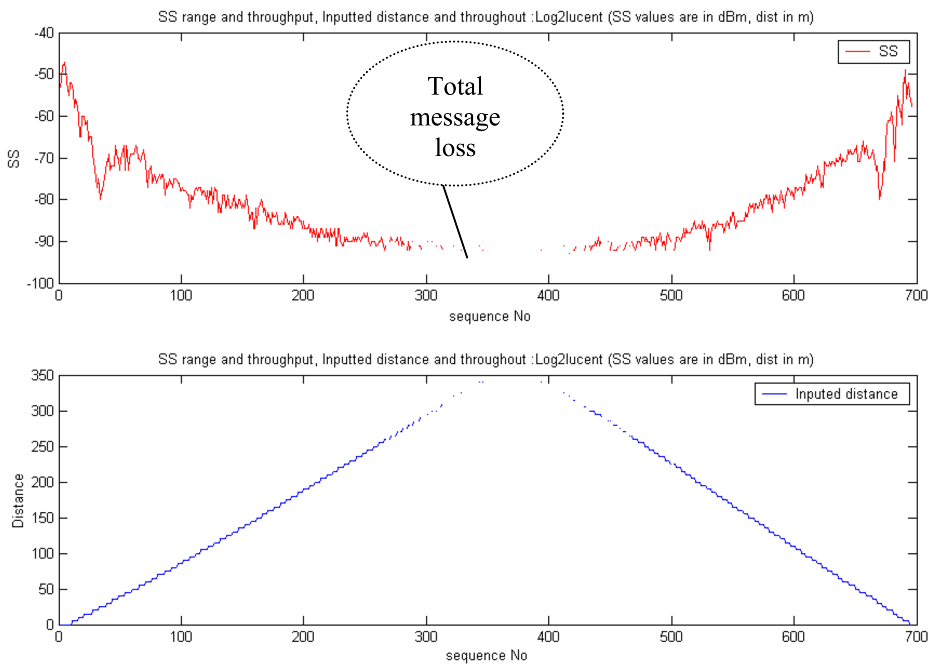


Figure 2, decrease in signal strength as distance increases

This graph shows the decrease in signal strength as physical distance increases. As noted in the previous section’s distance comparison, communication becomes impossible roughly around the 290 meters mark.

3.4.5.1.2.3 Comparison of signal strengths from different tests

The previous graph demonstrated the increase in distance leading to a break in communication. It also showed the steady decrease in signal strength leading to the break. The following graph shows the received signal strength at both the leader and client nodes taken over three tests. These three were repeats of the same test, taken to ensure that results were consistent. The received signal strength is on the Y-Axis, and the X-Axis marks frame sequence numbers. The signal strength values are in dBm.

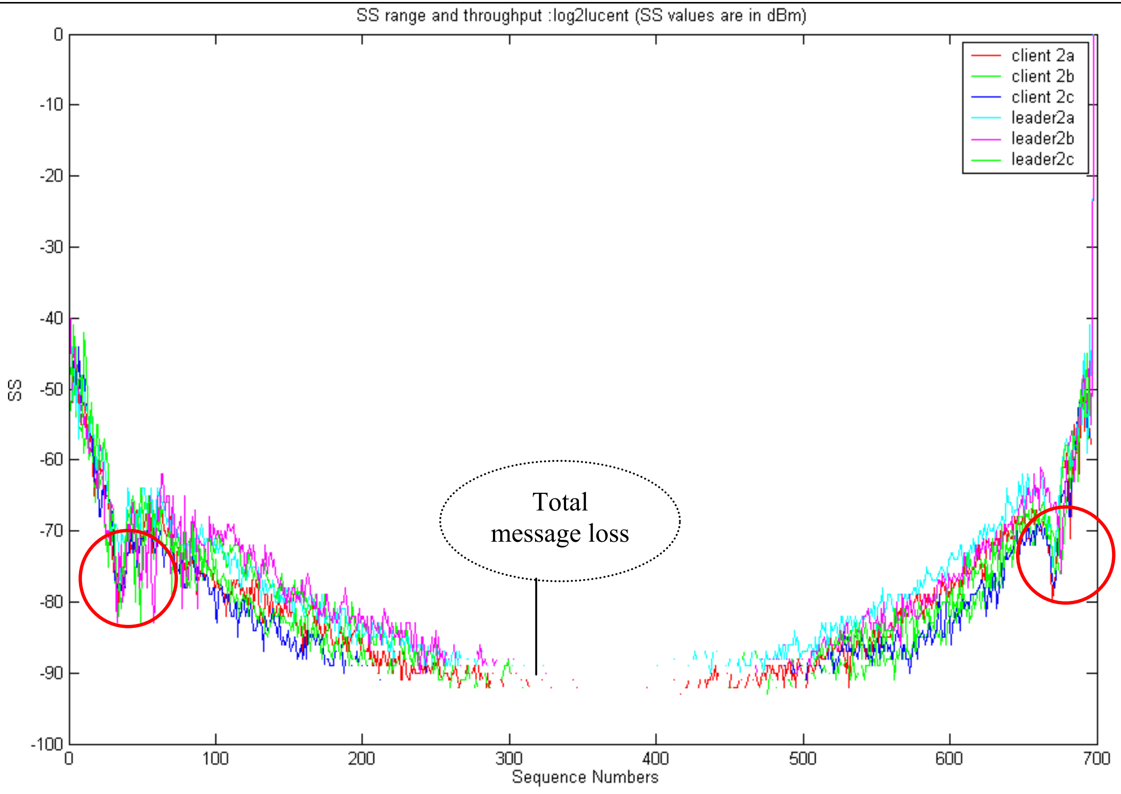


Figure 3, 6 samples taken across three tests

This graph shows a consistent theme across all 6 sets – signal strength will decrease until communication between the two nodes is no longer possible, then it will increase again when the moving node comes back into range. Communication becomes untenable when signal strength is between –80 and –90 dBm. Two other interesting qualities in this graph are the dips in quality marked by the red circles, and difference between the received values at leaders and clients in the same test.

The dips marked by the red circles are a peculiarity. At present, the best explanation that seems to fit is that the distance at which the they occur (when nodes are approximately 25 meters apart) is the distance at which the reflected signal bounces off the ground and arrives at the receiver, causing an echo effect with a direct signal. In any event, they are not vitally important, as they do not cause any messages to be lost, or the received signal strength to drop to dangerous levels.

The final noteworthy item in this graph is that, in a test with two nodes sending messages to each other, with the same set up, the received signal strength at their respective interfaces is consistently not the same. The following graph is the same as the one above, but with only one test described. It will show this occurrence more clearly:

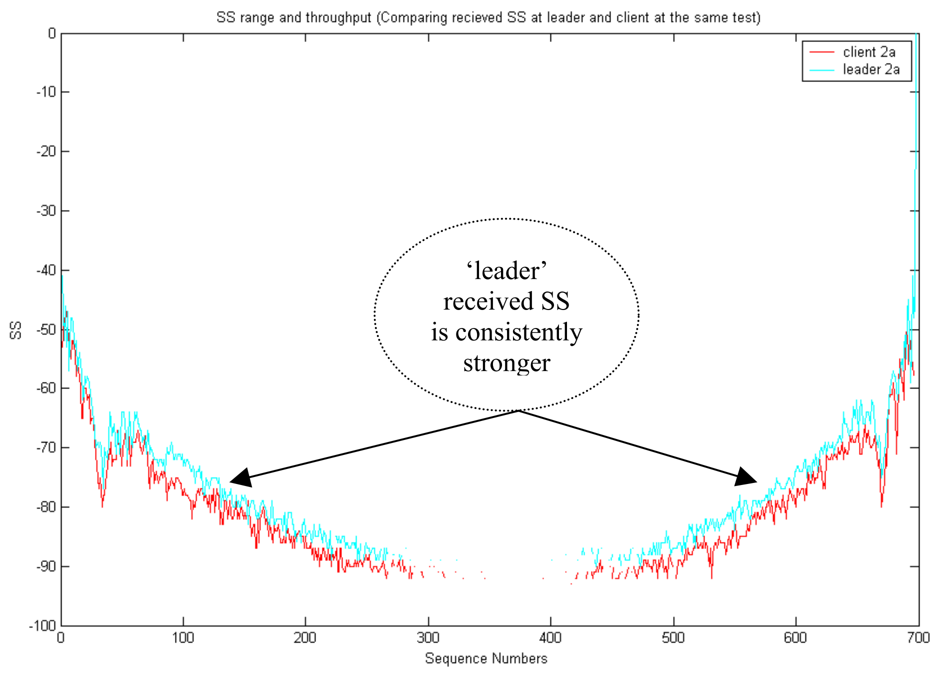


Figure 4, two nodes received signal strengths during a test

The received signal strength is consistently stronger at the leader node. One possibility that might explain this result is that the leader node is stationary, and the client node is moving at all times. This would imply that a non-moving node has a greater chance of receiving a message than a constantly moving one.

This difference in received signal strength leads to the observation that in a test with where two nodes have the exact same hardware and software set up nodes can receive different statistics about a connection. It does not matter if this is the result of one node being stationary or not, as in a wireless network the movement patterns of nodes cannot be assumed to be stationary or mobile.

This means that a connection between two nodes cannot be considered bi-directional, although the statistics for both directions could be very similar.

3.4.5.1.2.4 Compare distances achieved on sand to grass

The following graph shows the distances achieved before transmission loss on grass and sand.

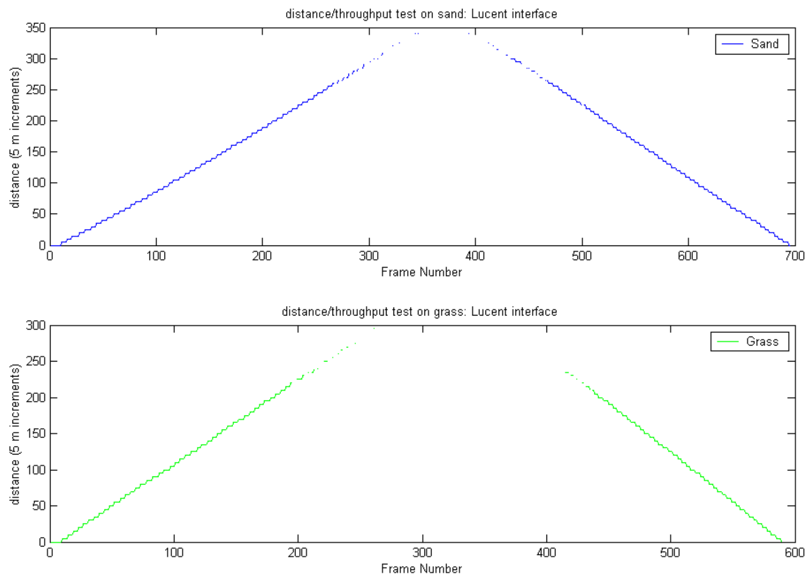


Figure 5, comparison of throughput on grass and sand

Major message loss occurred on grass at approximately 240 meters. On sand, as noted before, this figure was approximately 290 meters. This conclusion that has been taken from this is that grass surfaces adsorb electromagnetic waves better than sandy ones.

Comments

The two major patterns observed here are that signal strength is a good indicator of the possibility of communication failure, and that distance cannot be treated as an absolute value for inferring the possibility of failure. Other observations are that communication quality between nodes is not bi-directional, that different wireless interface brands have different abilities, and that environment will have an effect on communication, even in line of sight cases.

3.4.4.2 Category two: Influence of blocking on signal strength received

This category will discuss one of the blocking tests carried out during the course of the experiments. It is from the 'orientation of devices', and not the 'influence of blocking' category[9], but this test proved to be a crossover between the two. It is mentioned here as it has had a direct result on the effectiveness of a partition anticipator.

3.4.4.2.1 Experimental set up

In this test, testers would hold the two nodes 150 meters apart. The testers would face each other, and there was a line of sight between the two testers. During the test, the tester holding the client node would slowly rotate 360 degrees in a clockwise fashion, noting the degrees value every 90 degrees. A test constituted one full rotation. This test was performed three times to verify the result.

3.4.4.2.2 Experiment and results

The test involved cutting off the line of sight between nodes by having one tester rotate on the spot in a full circle, holding the laptop in front of the tester. The received signal strength at the client node for a test is shown below.

[9] The 'orientation of devices' category investigated if the PCMCIA wireless interfaces had a directional bias – if the transmitter on the cards was configured to send in any one direction in a more powerful way than in the other directions possible in the 3D environment. The 'influence of blocking' category looked at how the removal of line of sight affected transmission.

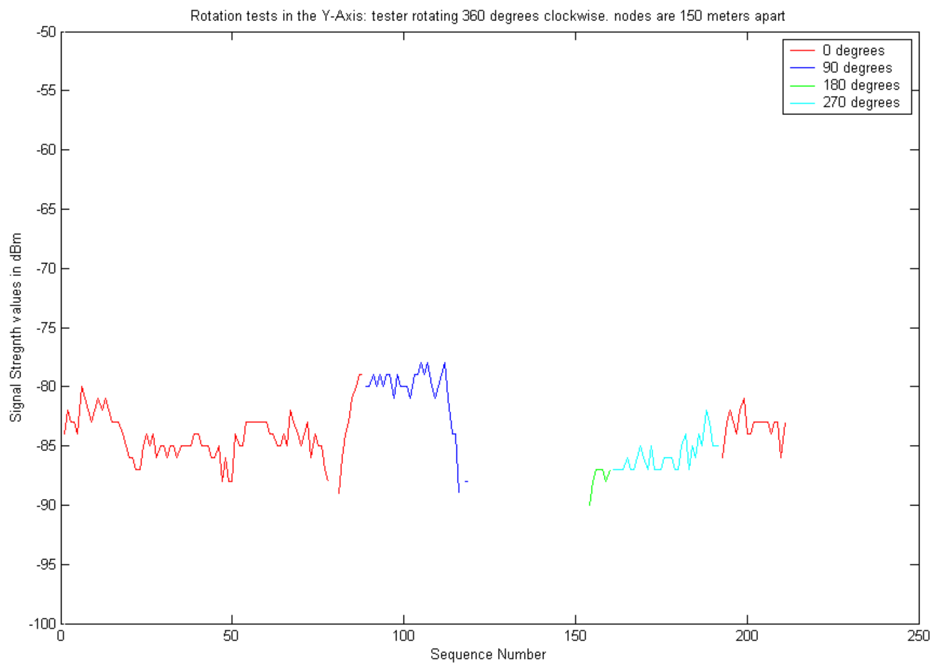


Figure 6, rotation in the Y-Axis

The received signal strength at the rotating node is already weak due to the distance between the nodes. Communication breaks down totally when the rotating tester moves 180 degrees and is facing in the opposite direction to the stationary one. The rotating tester has blocked the signal both ways (the same effect is visible at the leader node), and the nodes are unable to communicate, despite being within range. Link quality increases again as the tester returns to the starting orientation. This result correlates a similar pattern observed in the RADAR project, which is discussed in chapter 2.

Curiously, the link quality is slightly better when the tester is orientated 90 degrees from the other node, although not in any significant way. This was consistent across the three versions of this test, and lends credence to the notion that the wireless interfaces used have a directional bias.

Comments

This test is interesting as it highlights the effects of the short-range environment on wireless communication. A user holding a mobile node can turn around, and this can sever the link between two nodes without warning.

3.4.4.3 Mention: speed of sending and payload size

Two experiment categories that cannot be fully analysed due to space constraints, but that are worth mentioning are those that deal with the speed of sending, and payload size, of frames sent over the network. The following table shows the distances achieved until serious message loss occurred with different speeds of sending frames and different payload sizes for those frames. The figures in the table are approximate values, but they indicate some noteworthy trends.

	Fast sending (every 40 msec)	**Medium sending (every 100 msec)**	**Slow sending (every 500 msec)**
Payload is 100 bytes	Approx. 220 meters	Approx. 225 meters	Not available
Payload is 500 bytes	Approx. 225 meters	Approx. 190 meters	Approx. 210 meters
Payload is 2000 bytes	Approx. 160 meters	Approx. 160 meters	Approx. 190 meters

One interesting item here is that, for each different speed of sending, as message payload size increases, the distance until lost messages occur decreases. This indicates that frame size can be linked to the chances of successful transmission, and that larger frames have less capability by way of range.

On a lesser note, for the first two payload sizes, there is not any clear trend in different distances achieved with different speeds of sending. For the largest payload size tested, however, there is a difference of 30 meters achieved before loss occurred between slow and medium sending. This may mean that that speed of sending is a factor in distance achieved. However, there is not enough data here to draw that conclusion.

These figures indicate that speed of sending may be a factor in successful transmission, and that payload size with large frames is a factor. The wireless device, and not the 802.11b standard, could well be the cause of both of these results – as was the case in the blocking experiment presented above.

Comment

The above section mentioned two results highlighted from the set of experiments performed. The results presented are indicators only, and have not been conclusively shown here. They have been included here in an attempt to show some of the potential complexities that a wireless network analysis tool could encompass in order to accurately classify network conditions.

3.4.5 Conclusion of experiment descriptions

The descriptions provided have attempted to highlight some of the discoveries from the experiments carried out. The large amounts of data from the experiments have resulted in the necessity for brevity in this chapter, and so not all of the discoveries made have been presented. Also, the time constraints of this dissertation have prevented a full analysis of the data gathered; so many open questions remain to be investigated.

3.5 Chapter conclusion

The experiments presented in this chapter are a small subset of the entire set. They are enough, however, to provide enough understanding of the wireless 802.11b environment to design the LID protocol and a PA that can use it. The results presented will be used in the following chapters for the design and implantation of those programs.

Chapter 4

Analysis and Design

4.1 Introduction

This chapter will deal with the problem analysis and design for the partition anticipator and location information dissemination (LID) services. The chapter will begin by discussing requirements for those services in light of the preceding chapters, and then describe the intended implementation of those requirements.

The first section of the analysis / design will concentrate on the LID protocol, and the next on analysis / design for the partition anticipator. First, an open issue will be discussed – the placement of a partition anticipator on a network.

4.2 Should a partition anticipator be a centralised algorithm?

In chapter 3, the Teminodes project [11] was introduced. One aspect of it that was noted was that, at that projects group management layer, a group had a central member, whose movement dictated the movement of the group at large. Group management is beyond the scope of this project, but the idea of a centralised control point for affecting decisions among a physical set of nodes is worth considering. One question that has come up in the discussions about this projects version of a partition anticipator is where on a mobile network such an algorithm should be run.

The decision to be made is about the PA mechanism: should it be distributed, with a PA protocol running on each node and making local decisions, or centralised in a group structure, with the 'centre' of the group making decisions and informing the group members of its decisions. The arguments for these will be discussed now.

	Each node should have a local PA service	**PA service in a group, managed by a central node**
Chance of inconsistencies in group view between nodes	High chance – this is the assumed state	Low chance
Message count	Higher overhead	Less overhead
Time to notice a partition	Local scanning will find a local partition fast, but news of partitions between remote nodes will take longer to propagate.	Global news service, still has the problem that news of partitions between remote nodes has to reach controller before rest of group can learn
Point of failure	Nodes are autonomous.	Controller is a single point of failure in a group.

Comment

By controlling the PA service from a central point in a group, group structure is being forced onto the use of this partition anticipator. This is not a modular choice, as it ties this PA system to an aspect of the group communication paradigm. Also, as this project is not concerned with implementing other group management features in this PA protocol, such a choice would be violating the end-to-end argument[10] by implementing a pseudo-group management protocol.

[10] The end-to-end argument suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level. It is documented in [13].

Another argument is message count; the number of messages arriving at the application layer with a centralised group solution will be less - rather than all nodes beaconing messages to each other, all nodes talk to one controller and receive messages from it. However, the nature of communication in the chosen wireless technology (802.11b) is that any messages sent by a node are received at the MAC level by all nodes. It is a broadcast based technology. As such, the message count can be considered as being the same for both the centralised and distributed options.

The inconsistency factor is another argument worth discussing; with the group-centralised option, there is less chance of member nodes having an inconsistent view of the state of the network, and no chance of them making decisions inconsistent with those of others. However, a working assumption for this project about the nature of mobile environments is one of constant change. With this in mind, real time consistency between all nodes would not ever be possible by definition.

By one node choosing a network view and propagating that view to all nodes, there is the feature of a global group view at a known time on all nodes in a group. However, a full group management protocol could achieve the same functionality at a higher layer.

The last argument is that a controller is a single point of failure. An election agreement is required for a new group or in the event of partition / failure. The centralised choice necessitates an algorithm to ensure consistency following a failure.

With the arguments above in mind, the distributed option has been chosen. For a chaotic and fast moving mobile environment, it has been deemed better to distribute the PA system, and make each node autonomous from other nodes in the network. Each node will model its own network graph in a location information dissemination protocol, and will hold its own partition anticipator protocol to make predictions based on that network view. A feature for out of LID communication will be provided with the facility that messages can be piggybacked onto LID protocol beacons. Other processes on a node can use this to communicate between nodes, and, for example, make agreement on a group view.

4.3 Analysis of this LID protocol

Requirements for a location information dissemination protocol

In previous chapters the features of a LID protocol have been outlined, both in terms of goals and of common features of similar protocols.
The basic needs from a LID protocol can be defined as the following:

Functional requirements

- Input for a partition anticipator in the form of current network data.
- Network devices to learn about each other, and about the quality of the network.
- A means of disseminating information around the network.
- Be able to provide an estimate of a node's coverage on the network.

Non-functional requirements

- Efficient beacon communication
- Run with a minimal thread model

The functional requirements can also be broken into several implementation areas: the transport mechanism to be used, the interfaces to that transport mechanism, the information that a device will hold about itself and the network, and the intelligence that the device applies to its network view. The non-functional requirements will be used as guidelines throughout the design.

Each node will maintain its own view of the network using the information propagated by the LID, and will run its own version of a LID protocol. Each node running the LID protocol will periodically distribute a beacon of its current state and the state of its neighbors as it has seen them. There will have to be some degree of trust/timing associated with incoming information, as the beacons content will be inherently out of date. Killijian et al [7] advises setting a hop count (L_{max}) in order to prevent the propagation of information beyond a point where it is almost certainly stale. This hop count should grow and shrink according to the quality of the network, using intelligence in the LID protocol. A figure for the periodic beacon rate will be determined during the implementation stage.

4.4 Design of the LID protocol

4.4.1 Pre-requisites for this LID protocol

Three requirements that the design has of its underlying hardware are that it:

- Be able to poll its network interface for information about the other nodes in the area.
- Be able to send and receive data directly at the frame level, without reference to higher layers of the networking stack.
- Be able to poll a network interface in order to find out the MAC address that the interface is using.

These features are provided for by three extensions to the network interface driver. These extensions will be described before moving onto the design for this LID protocol.

4.4.2 Sending and receiving data without IP [16]; the work of the DSG, Trinity College Dublin

The Distributed Systems Group of Trinity College Dublin have developed extensions for the 802.11 open source driver that allows for sending data to and receiving data directly from the MAC level. The extension is useful for this projects needs as it allows for complete control of a frame payload, and it removes the possibility of a beacon becoming fragmented while it is being sent by the LID protocol, and as.

The driver is written in ANSI C over Linux. It provides functions that wrap around specially designed system calls. These accept a pointer to a C char data type array for sending, and return a pointer to a char array for receiving. A kernel recompile is necessary to install the custom system calls.

Fragmentation of beacons is not an attractive option, as it leads to increased overhead in processing. In the MAC receive case in the LID protocol; for example, complexity requirements would increase from having to hold sections of a fragmented beacon, or having a timeout to wait / block for the rest of a beacon to arrive. In the general case, by allowing fragmentation, the chances that a beacon will only have half arrived are increased– that some of a beacon message will be lost, rendering the entire message unusable.

It is an important point that all received frames are copied to a circular input buffer, rather than queued for polling in the main driver code and then discarded once read by a program polling the DSG extensions for any received frames. This way, other programs on the same machine can still use the network interface without any observed change in functionality. Reading a frame from the DSG input buffer, however, will cause it to be removed from that buffer. Programs polling this buffer can either block waiting for messages or choose not to wait if there are no messages.

4.4.3 Network monitoring: collecting statistics about other nodes on the network

It is useful for this LID protocol to be able to record information about the other nodes on the network. To this end, the wireless driver was modified again, this time in an unrelated area to the DSG changes.

In the new version, statistics are recorded into a data structure in the driver every time a new frame arrives into it. This data structure has a fixed maximum size, and new data wraps around and overwrites the oldest data in the structure when it comes in. This size limitation was to ensure that there would not be any appreciable drop in driver performance[11] that might result from the card holding too much data. A socket is opened to the device driver in order to poll, and statistics are returned.

Structure of statistics values:

For each place in the data structure the following data is recorded:

- MAC address of observed node

[11] The Linux wireless extensions [15] have made a similar addition to the driver for their iwspy tool, using the same type of data structure to different effect. These extensions set a maximum size for that data structure as a value of eight, meaning that statistics can be held on a maximum of eight network interfaces. The extensions implemented for this project observe the same value.

- Signal strength frame was received at
- Quality frame was received at
- Noise level on network at time of frame arrival
- Whether this reading has been polled before

For each set of values that has not been seen before by the LID protocol, the values are passed to be added onto the relevant place on the network graph. If a set of values describes a node that is not in the network graph, then that set is not used. For this LID protocol, only nodes that are also disseminating beacons are interesting. If that particular transmission had been a beacon, then the node that it had come from would have been added to the network view by the time this functionality had been called.

Values returned by the wireless interface

The figures reporting on the quality of a wireless transmission are: the signal strength that the signal came in with, the noise on the network when it came in, and the difference between the two – the quality figure.

Depending on the brand of interface used, these figures are returned either as dBm, or as relative measurements compared to the maximum range of the wireless interface. This range value is collected from the wireless interface. Using a general value that can be understood by all nodes irrespective of the abilities of any wireless interface used is a goal for heterogeneity. In four out of the five card tests that were run, values returned were in the dBm scale. However, in one of these tests, using a Compaq card, values were returned in the relative scale. This is not of use to a LID protocol– globally meaningful values need to be reported for network dissemination.

In order to remedy this problem, absolute (dBm) values need to be extrapolated from the relative scale values returned by interfaces like the Compaq. Inferring these values on a manufacturer-by-manufacturer, or card-by-card basis is not a scalable solution, as there are many different wireless interface brands and products. As a result of this, an analysis needs to be done that will study the values returned by several different brands of cards, so that a generalised framework for mapping relative values returned to the dBm scale can be inferred. This would not be an onerous task, but it has not been completed in this dissertation due to time constraints.

Comments

An alternative approach to the need for information about network nodes would be to modify the driver to return this sort of information with each frame that is received from the DSG input buffer. While this solution would have achieved the same goal, it would have curtailed the ability to collect statistics, meaning that only information from frames collected by the LID protocol could be used.

By solving this problem with this type of driver change, a feature has been developed for all inter-node traffic quality to be monitored, be it intended for the LID protocol, for a higher network layer on the same machine, or for another network node entirely. Also, there is a decoupling of the functionality of collecting network statistics from the LID functionality, and a tool has been created that can be used by other applications for other purposes.

Through testing in a busy college wireless LAN, a rule of thumb has been developed that polling roughly once a second is a good rate for getting new statistics with each poll. This number is highly environment and network traffic dependant, however.

4.4.4 Getting the MAC address of an interface

Linux offers programs that can get the MAC address of an interface. However, their use would leave the LID protocol reliant on external software. Because of this, a small change was made to the wireless drivers to return the MAC address of the mobile nodes wireless network interface on demand.

4.4.5 The LID protocol: aspects of design

This section describes the key aspects of the design of the LID protocol, followed by a more detailed look at the layout of the LID program design itself. The LID protocol has two main functions:

- Maintain a network graph that is pruned periodically.
- Send and receive beacons that access and add to that network graph.

These will be described now.

4.4.5.1 Description of a network model

Each node holds a model of the network compiled from received beacons and from statistics polled from the wireless interface. The components of a network view will be described now.

The network view is logically viewed as a connectivity graph. Network nodes are modelled as nodes on the graph, while the connections - 'links' - between them are modelled as edges.

4.4.5.2 Discussion of 'edge' modeling

It is easy to conceive of models of use in wireless communication where one node has a more powerful receiver or transmitter than another. A homogeneous node ideal is limited in scope as it means the assumption of the same hardware, using the same configuration at all nodes. Different brands of wireless interfaces have been observed as having different capabilities in terms of range, and perception of the network. Also, in the chapter reviewing the 802.11b technology, it was shown that between two nodes of the same configuration the signal strength received at one node was often different to that of the other. Link quality between two nodes, using the same hardware and software set-up, is not bi-directional.

For the above reasons, graph edges (the connections between nodes on a network connectivity graph) will be modelled as unidirectional. Each pair of nodes that are connected will have a pair of unidirectional links associated with them, with a link's starting point 'belonging' to the node from which it originated and referencing the terminating node. A link models a transmission sent from its owner and received at a destination. When a transmission is received, the recipient will record data about the conditions under which it arrived, and use these to model the link from sender to recipient. The signal strength and the quality recorded on the network will be used for this.

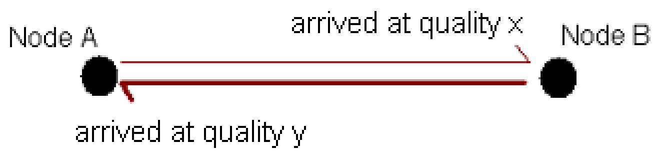


Figure 7, notion of a pair of unidirectional links

A sender will learn about how other nodes received its transmissions when it gets a receiver's beacons.

Over time, a history can be built up of connections between nodes, which can be used to track change.

4.4.5.3 Modelling a node

A node and its links to other nodes is modelled as the following collection of fields:

- **Address**
- **Hops to root of this graph**
- **Last sequence No**
- **GPS history**
- **Extra information**
- **Last time modified (local time)**
- For each link:
  - Address
  - Quality history
  - Last time modified (local time)

The motivation behind each field is explained now.

Address: this serves as a unique identifier for a node. For the purposes of this project, the MAC address of the wireless network interface is used.

Hops to Root of this graph: this is useful in pruning a network graph. It is re-calculated periodically

Last sequence number: when information is received about a node in a beacon, it is useful that the node that it describes have attached a sequence number to it. This is used in order to prevent old information overwriting new.

GPS history: GPS is an optional feature in this LID protocol. This is due to a desire to use the LID protocol in environments where GPS is not feasible. This field holds a history of reported GPS movements for that node.

Extra information: This is a facility for out of LID communication. Should a higher layer protocol wish to piggyback information onto the description of a node in a beacon, the piggybacked information transmitted will be stored here. One example of optional data that a higher layer might want to include could be synchronization messages.

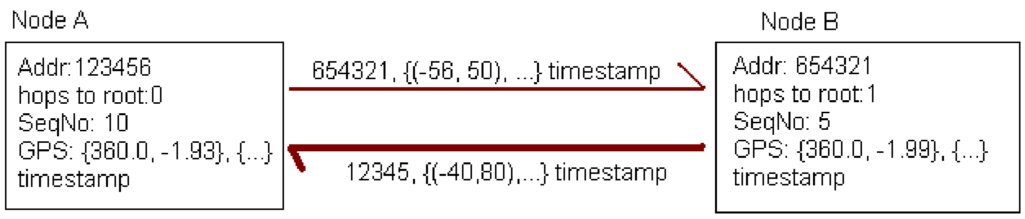
Last time modified: this field holds the timestamp time that information on this node, or any of its links was updated. The timestamp is held in local time and is not propagated with a beacon.

Links: It has been stated that a link from one node to another is thought of as belonging to the sending node. With this in mind, a node data structure also holds a collection of link structures. A link is made up of three fields, which will be described now.

> **Address**: an identifier for the destination node. This is the MAC address of that destination.
>
> **Quality history**: the quality of a link is measured in two fields: the signal strength that a transmission was received at, and the quality of the network when it was received.
>
> The figure for quality is derived from the signal strength and the noise recorded on the network, so the noise does not need to be recorded as well as

the other two pieces of information – one can be inferred from the other. Link quality information is kept in a history data structure, so that a LID protocol can show change over received links information.

Last time modified: this is a timestamp value in local time, recording the last time that information on this link was added to.



Node A
Addr:123456
hops to root:0
SeqNo: 10
GPS: {360.0, -1.93}, {...}
timestamp
654321, {(-56, 50), ...} timestamp
12345, {(-40,80),...} timestamp
Node B
Addr: 654321
hops to root:1
SeqNo: 5
GPS: {360.0, -1.99}, {...}
timestamp

Figure 8, diagram of a network graph held at node A

A network graph will be made up of a collection of nodes and the links between them. The root node of this graph will describe the network device that that graph is held on. This root node will hold the normal node fields, plus some extra pieces of information. The extra fields that will be held are:

- **Type of antenna**
- **Interval between beacons**
- **Value for L$_{max}$**
- **Figure for the average noise at this node**

These extra fields are used to run the LID protocol. The first field - type of antenna – is not used at present. The "interval between beacons" value holds the sending rate for beacons. The L$_{max}$ value is used to prune the network view – as a boundary value against stale information. The figure for average noise at the device is kept as a statistic value for a PA system to use.

4.4.6 Pruning the network graph

Periodically, the network graph will be pruned in order to keep it within L_{max} size. Constructing an OSPF routing table using the root of the network tree as a starting point will do this.

The hop count in the generated table will be used to cut off nodes that have a greater than L_{max} distance from the root node. A decision was made that in mobile networks an expensive (in terms of signal strength) single hop is more reliable then a cheaper one made up of several hops. As a result of this, hop count was chosen as the metric for the OSPF table generation instead of aggregate signal strength. This section of functionality also resizes the L_{max} figure by analysing the size of the pruned graph as opposed to the pre-pruned graph.

4.4.7 Beacon construction and contents

4.4.7.1 Overview

The motivations and design choices for a beacon will be discussed. A beacon will be contained in the payload of a single 802.11 MAC frame. This size limitation means that the contents of a beacon will have to be made as small as possible in order to fit as much information as possible into a beacon. A heuristic value for the size of an 802.11b frame, balancing biggest size without too much throughput loss for the tested[12] environment, has been chosen from the 802.11 tests. This heuristic value is 1156 bytes, which is half of the maximum size of an 802.11b MAC frame (2312 bytes).

Beacons are sent on a one-hop basis – a node will receive a beacon, absorb its contents, and send its own beacon, but it will not forward a received beacon.

4.4.7.2 Beacon contents

A beacons purpose is to describe the network graph held at a node in a compact fashion, so that as much as possible of that graph can be sent. A beacon will

[12] Tests were performed in an outdoor, line of sight open space on two surface types. See chapter 3 for more details.

be made up of an enumeration of the nodes and links in a network graph. The beacon is constructed in the following fashion:

- A header: describing the beacons contents.
- The root node, a description without its links.
- Each other node in the graph, including a description of its links.

The links that belong to the root are taken in from received beacons. Therefore, this information has already been propagated to the network, and is still being propagated by all nodes in the area while the root remains active. Because of this, a device does not re-propagate the links information corresponding to its position on the graph (the root node).

4.4.7.3 Breakdown of the beacons contents

A beacon is made up of a header, then an iteration of nodes. Information is encoded at the binary level, as it makes the transmissions more efficient in terms of size. Also, by using custom encoding for the various data types that are being encoded in messages, some problems that could occur when running the LID on different hardware architectures[13] can be ignored. A beacons structure is described in more detail now.

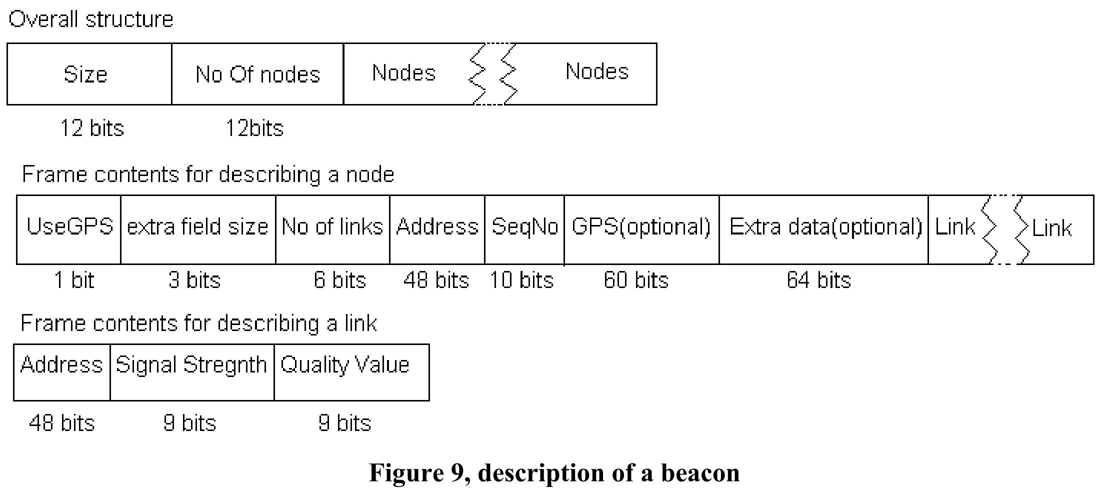


Figure 9, description of a beacon

[13] One example of a problem that could occur in different hardware systems would be the different memory allocations given to the data types that are placed in a beacon. By encoding in a custom code, this possibility is removed.

4.4.7.4 Overall structure

The first 3 bytes of the beacon (24 bits) describe the overall size, and number of nodes described in the beacon. The overall size figure is used to eliminate malformed or non–LID protocol frames from being read. A maximum sized 802.11b frame can hold 31 node descriptions, where each node has 6 links associated with it.

4.4.7.4.1 Node description

There are three pieces of information that are of variable size in a node description. These are the space allocated for GPS data, the space allocated for extra data (out of LID communication data), and the number of links that this node has. The first three fields of the node descriptor hold information on the use of these three variable fields.

- USEGPS: if GPS co-ordinates are included or not in this description.
- Extra field size: describes the size of the extra data field – a value between 0 and 7 inclusive.
- No of links: describes the number of links that follow this node descriptor before the next node descriptor. Note that this field allows for a maximum of 2^6 (0 to 63) links.

The first 68 bits of a node description are fixed, and depending on GPS data (60 bits) and extra data (64 bits) optional use, there could be as much as 196 bits describing a single node before any links data. The composition of the rest of a node description is a follows:

- Node network address.
- Sequence number:
- Of current message in the case of a root node descriptor, and of last received message in the case of other node descriptions.
- GPS data (if being used).
- Extra data (up to 8 bytes extra data can be included for out of LID protocol communication).
- Links.

4.4.7.4.2 Link description

There are three pieces of information encoded in a link. These are:

- Destination address.
- Signal strength.
- Network quality value.

Each link is fixed at 66 bits in size.

4.4.7.5 Comment on sending network quality figures

The figures for network quality (signal strength and noise) are reported in the dBm scale. This means that the values returned from the wireless interface are in the minus scale. As these values will always be in the range of 0 to –100 (as observed from the wireless tests), the minus value can be stripped from the dBm values and the values sent as positive integers between 0 and 100. This saves some beacon space that would have been taken up by redundant information.

4.4.7.6 Creating a beacon from a network graph, parsing an incoming beacon

Beacons are created from network graphs. The root node is added, then each node in iteration with its links.

In the case that there is a large network graph to represent, there is the possibility of a problem in that there might not be enough room to fully represent all nodes and their links. In this case, preference is given to representing all nodes before representing any of their links. This is based on the premise that it is better to know about the existence of nodes than it is to learn about the connections between them.

Once all nodes have been allocated for, if there is still space left over, each node is allocated a quantity of space for its links. If a node has more links than it has been allowed space for, it will send the 'best' (in terms of quality) links that it has. If a node does not use all of it's allocated space, then the leftover is added to the quota of the next node to be processed.

4.4.7.7 Process of marshalling and un-marshalling beacons

A data structure to hold binary values is created. For each node to be added, the relevant data fields are converted to network order (big endian) binary with the trailing zeros stripped for brevity, and added to this data structure.

As previously mentioned, the DSG wireless driver has been programmed in C on Linux. As payload contents it takes a pointer to a C char data type. As a result of this, the frame payload has to be presented to the driver as an array of characters. To this end, once all the relevant data fields have been encoded and added to the binary data structure, it is converted to an array of type char using the ANSI character set, which is then passed by reference to be sent.

Decimal Values				
Use GPS (1 bit)	Extra field size (3 bits)	No of links (6 bits)	MAC Address (48 bits)	other data..
0	7 bytes of info	10 links	00:02: …	…
Binary representation				
0	111	010100	00000000:01000000:..	…
Binary values together				
01110101000000000000001000000……				
Representation as individual characters: one character == 8 bits				
01110101	00000000	00010000	00 …	
u	NUL	DLE	…	

Example of encoding of data

Once the data to be encoded has been represented as an array of characters, it can be sent as payload in a MAC frame. The size in bits of a single character is used to break down the data structure for encoding. As a result of this, should a different system upon which the LID protocol be running encode the C char data type in a memory size different to the standard eight bits, the system will still be able to run with no change in functionality or behaviour.

In the event that the end of the data structure holding the binary values has a remainder less than the memory allocated for a single character, it will be padded with zeros that will be ignored at the receiving end.

Translating back from this format is done in much the same way. The received payload, an array of chars, is converted to a binary representation in a data structure. This data structure is indexed and the relevant sections of binary converted into values used to propagate various nodes and links data structures. These structures are then grafted onto the network view of a receiving node.

Comment

The procedures for modelling network graphs and creating beacons have been looked at. Next, an example of a two-node network will be shown. This next section demonstrates nodes sending beacons to each other and learning of each other's existence.

4.4.8 Example of LID learning process

A link from one node to another describes how a transmission, sent from a node, was received at a destination. This information is added to a network view at the receiving node. The sending node will learn about how its messages were received when it gets a return beacon from its neighbours. The next section will describe a typical LID protocol learning process.

In the diagram below, suppose A starts sending and sends an empty beacon (A has no network information yet). The beacon arrives at B, who absorbs its contents, and then records that there was a transmission sent from A that arrived at B at quality x. B can then include this information in its graph, and also in its next outgoing beacon.

B's beacon arrives at A, is parsed and a figure for the transmission B -> A can be added to A's network graph.

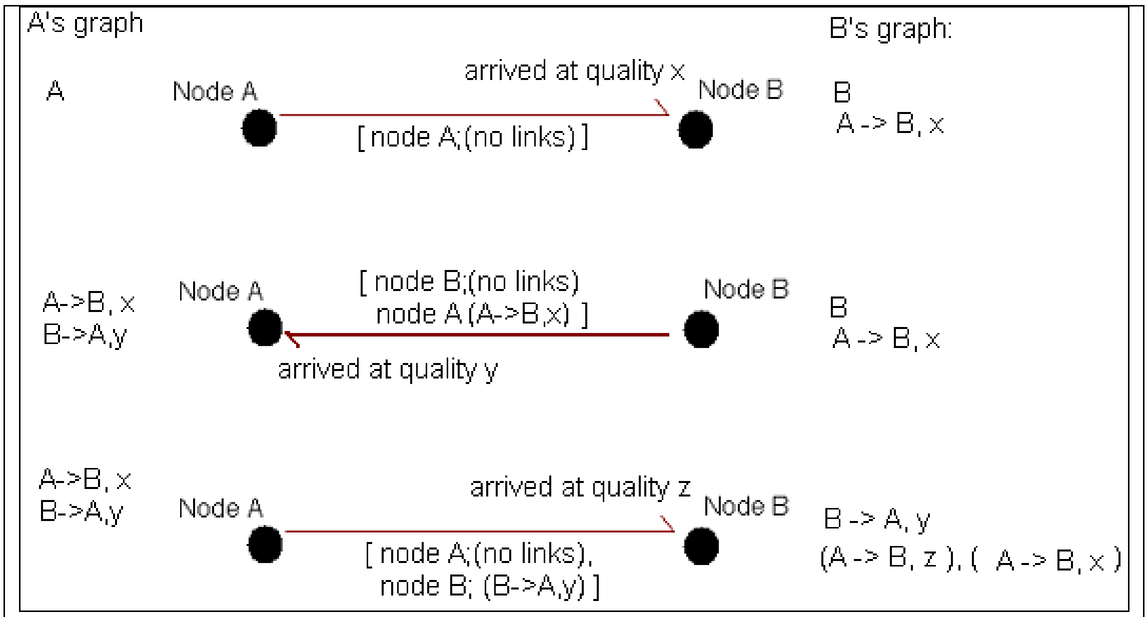


Figure 10, transmission of beacons between two nodes, build up of history

In the final, third beacon, A describes itself, and also the figure for the quality of B's last transmission to A. Once it has arrived at B, B now has a figure for how its last transmission to A was received, and also a new figure for the transmission from A to B.

4.4.9 Design layout

Now that the key aspects of the design for this LID protocol have been outlined, the design diagram can be shown. The following structure will be used for implementation:

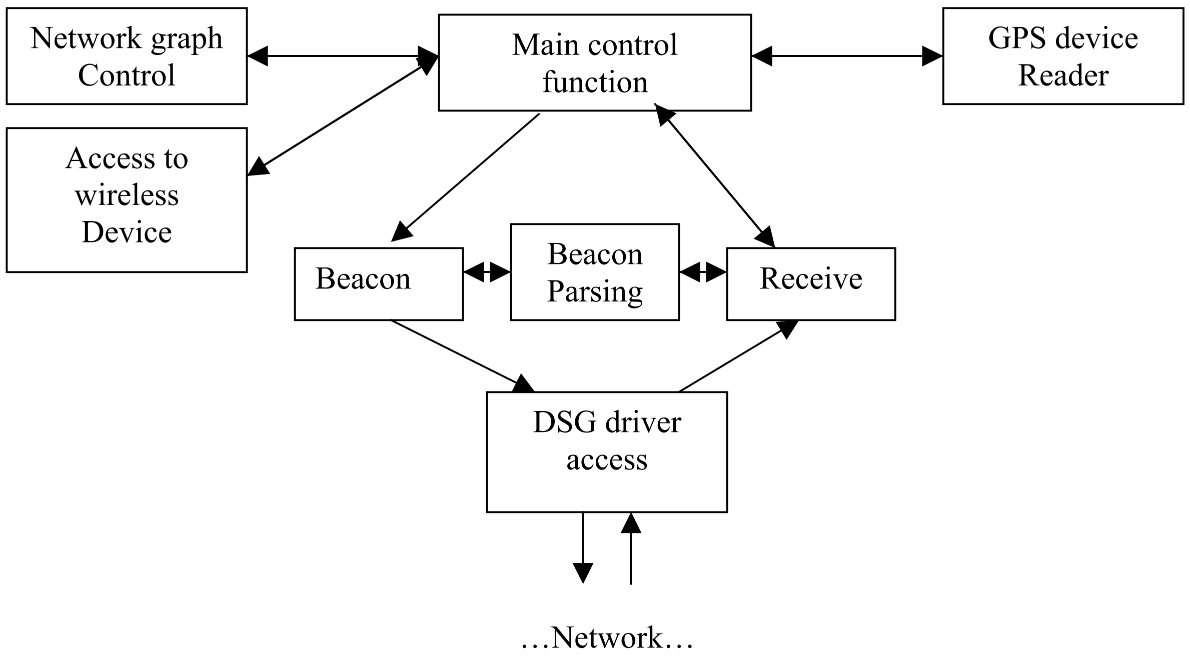


Figure 11, layout of the LID protocol functionality

The following function modules can be defined.

Network graph module

This module is used to model and maintain the network graph structure. Features include adding and removing nodes and links.

GPS device reader module

This module is concerned with accessing a GPS device, and providing figures for GPS location.

The implementation of a serial port reader that parsed GPS sentences will be taken from the GeoDoom project [14] for this function.

Access to wireless device

This module is concerned with accessing the network interface to poll for statistical information, including the range of the wireless interface.

Recall that the wireless driver extensions returns data structures comprising five pieces of information per frame received – the MAC address of the sending interface, the background noise at the interface, the signal strength that a frame was received at, the general quality of the network at recording (the received signal strength minus the noise when that frame was received), and if this record has been returned before. When a node receives a beacon, it can then poll the driver and receive statistics about how that beacon was received. After adding the contents of a beacon to the network view, the information about how that beacon was received can also be added to the network graph.

DSG driver access

This will act as a wrapper to the DSG driver functions for sending and receiving at the MAC layer.

Beacon functions

The beaconing module will use the transport functions to send its beacons, in a push-based solution. When prompted, the beacon module will use the beacon parsing functions to generate a beacon, and then push it onto the transport functions to be sent. The working assumption here is a mobile network where there is no beforehand agreement or protocol about nodes sending beacons at specific times. As such, in this version of the LID protocol, nodes can be guaranteed to queue to send their beacons at known times but transmission of these may be delayed by the characteristics of the network or the wireless interface.

Receive functions

The DSG driver will hold received frames in a circular input queue until a receive function polls for them. Polling is non-blocking in this case – so if there are no frames to be taken in from the driver, then the LID protocol will not wait for there to be frames available. Received frames will be parsed using the beacon parsing functions to see if they are legitimate beacons. If they are, then they will be added to the network graph using the graph function set.

Main control function

This function set is the controller of all the other sets. It initialises the various devices used by the protocol, and starts the beaconing process. It also implements the network graph pruning functionality via the network graph module.

4.4.10 Conclusion of LID protocol design

The requirements and design for this location information dissemination protocol have been discussed. A design has been proposed, with the key features explained in detail. A design diagram that outlines the desired functionality and implementation structure has been provided.

4.5 Analysis of a partition anticipator

4.5.1 Introduction

In previous chapters examples have been shown of partition anticipators, and goals have been outlined for such a service. Generally, a partition anticipator should be able to report 'interesting' behaviour. For the purpose of this project, 'interesting' is behaviour likely to cause extremes of network quality: either very bad or very good.

The observed characteristics of the wireless technology used are based on outdoor, line of sight tests. If only these results are used as a basis for making decisions about quality, this version of a partition anticipator would only be really predictable in that environment. It could be easily conceived that a user might want to set their own rules for partition anticipation, or redefine the existing ones for a different granularity or functionality. With this in mind, one of the design goals for this partition anticipator has been to implement it in a flexible and modular way. Also, attempts have been made to separate the partition anticipator specific functionality from the inference functionality.

The requirements for this partition anticipator will now be described.

4.5.2 Requirements for a partition anticipator protocol

The needs of this PA protocol can be described as the following:

Functional requirements

- Take in data from a network information algorithm and use it to analyse the network.
- Anticipate immanent failure and issue a report about it.
- Recognise that a node is no longer sending beacons and issue a report about it.
    - This could mean that either the node has failed, or it is not able to communicate because it is out of range or its transmissions are being blocked. In any event, the PA system should notice and report it.

Non-functional requirements

- Model network quality heuristics in an easy to understand and efficient way.
- Allow for the easy adding / editing of heuristics, as new conditions leading to report-worthy states are defined.

4.5.2.1 Comment on functional requirements

The requirements here are pretty straightforward. The characteristics of the chosen wireless technology that have been observed from tests can be used as heuristics for analysing the LID gathered data.

Mention has been made of a partition anticipator reporting discoveries, but no mention has been made of how or to where reports would be made. Another protocol is envisaged as using the output of a partition anticipator in order to make some decision (for example a group manager deciding to cut a node from a group, or an emergency range warning deciding to inform the user, etc.). For the current purposes of the PA, it is sufficient for a partition anticipator to deliver its reports to screen. The evaluation and implementation chapters will address some other possibilities for delivery of such reports.

4.5.2.2 Comment on non-functional requirements

The non-functional requirements present a less concrete set of goals to work towards than the functional ones. It is very likely that the requirements for a PA system will change over time, and that conditions of use which can be viewed as fixed now will certainly not be so in the future.

4.2.3 General comment

The intelligent modelling of a decision-making framework is a key issue for design. From the wireless tests there is the notion of a set of condition-based checks that can be performed on various pieces of information that the LID protocol holds. The results of these checks will be indicators of the current quality of the network, and can be used to infer future quality. The problem is thus: how to model a set of variable based conditions in a dynamic fashion? An approach for rule-based inference will be discussed in the next section

4.2.4 Expert systems and knowledge bases, state machines

Expert systems are programs that mimic expert human opinion. They take inputs and apply them to rules until an answer has been reached. A provided input will be iterated over a series of rules until there are no more rules to apply. The last applied rule determines the result. An expert system uses a 'knowledge base' of objects and the conditions attached to them in order to model rules that an input can be tested against. The condition associated with an object in a knowledge base determines if the input matches that object, or needs to be further defined as being an object of another type.

As such, expert systems are similar in concept to the traditional finite state machine[14]. One can define a set of states, and have progression from one state to another based on the meeting of some rule defined as associated with that state. In this context, an object in a knowledge base is a state in a state machine, and its outputs to other objects are connections to other states in the state diagram.

4.6 Design of a partition anticipator

This section will discuss the design of the partition anticipator developed for this project. There will first be a brief introduction to finite state automata.

4.6.1 Design of a state machine for decision making

This partition anticipator will be designed as a finite state machine.

A state will have a single defined input, a specified state descriptor, a condition deciding whether to move onto possible output states or not, and two possible output states. The machine will be run until a state is reached that has no

[14] A finite state machine is a logical device that stores the status of some variable at a given time, and can operate on some input to change the status of that variable. Input can also cause an action or output to take place for any given status change.

more following states, or until the output from a state is the same state again – meaning that the input has been classified.

For example, in the diagram below, a very simple state machine has been drawn that classifies inputted signal strength into one of two possible values.

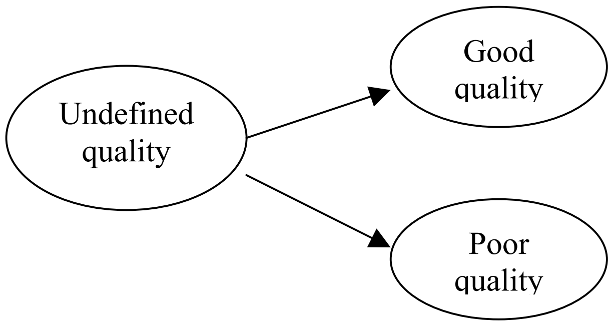


Figure 12, a simple state machine

A value is in an "undefined quality" state. From there it can move to a state of "good quality" or "poor quality". Once it has been defined the state machine is finished.

Using the rules in this example, the decision maker will take as input a signal strength value, and decide using its knowledge base what quality range that value belongs to.

4.6.2 Generalisation to a decision making framework, and a controller to create, request, and react to decisions

This partition anticipator will be able to take in values from the LID protocol as input, and use a state machine model of rules in order to infer results that can be reported.

If a PA system were designed that worked directly with the data structures from this LID protocol then it would only be functional with the LID protocol, or another protocol with the same data structures. Also, designing a PA system that can only be used to infer decisions about 802.11 type networks would limit the possibilities for a PA.

Instead, a breakdown of the PA system could be done, into a general-purpose inference framework, and a partition anticipator to control that framework. The general-purpose framework can model inputs, rules, and results to those rules, and the PA can create and use instances of the framework.

This would mean that partition anticipation becomes two separate functions. One will implement PA specific work, and the other will be an expert system model that is used by the first. This split makes for a more modular design.

Comment

By using a state based system to model the rules for this inference-making framework, the non-functional requirement for simplicity is met – A state machine is easily edited, and is modular in that is can be broken down into sets of small, workable state changes. These sets can be modelled and changed without reference to each other.

4.6.3 Design of inference-making framework

The inference-making framework is an implementation of a state machine. A state object can be modelled as a collection of the following fields:

- An attribute that the condition compares an input against
- A condition / rule to compare an input against an attribute
- A flag denoting the state of this object
- References to two possible output states

An attribute field will hold a value. A rule will be a simple function that can compare the attribute value and the inputted value. The rule will return one of two possible outcomes, and the outcome returned will dictate which state the machine is to move to next. If there are no more states, then the machine is finished.

Some things worth noting are that generally, rules will be very simple and taken from a set of generic rules used by most state objects. Also, an attribute should not be limited to being one data type – a framework such as this one should support several different types of comparisons.

4.6.4 Design of this partition anticipator

The partition anticipator will act to

- Model some knowledge using the expert system built.
- Get data for input from the LID protocol.
- Pass that data into the expert system.
- Receive the output and report it.

This partition anticipator is fairly simple in scope. Outputs will be printed to the screen. Two other possible output destinations could be a system interrupt or an IPC call to another process.

4.6 5 Conclusion of PA design

A design for this PA protocol has been presented. The functionality has been divided into a general expert system framework, and a PA specific controller that uses it.

4.7 Chapter conclusion

The requirements and design for the LID protocol and partition anticipator have been presented. Where applicable, functionality has been broken down into smaller functional sets.

The next chapter will discuss the implementation of the designed programs.

Chapter 5

Implementation

5.1 Introduction

This chapter deals with the implementation of the location information dissemination protocol and partition anticipator. There will also be a brief discussion of the extension to the 802.11b driver implemented for this project.

Development was done in standard ANSI C on the RedHat 7.3 operating system. C was chosen for its execution speed and efficiently compared to other high level languages. A Linux distribution was chosen for development because of Linux's open source nature, and ease of device access.

5.2 The 802.11 driver extension

The starting point for this extension was the open source PCMCIA WaveLAN drivers (Orinoco 0.11), developed by Jean Tourrilhes [15], and modified for MAC level access by the distributed systems group (DSG) of Trinity College, Dublin. These drivers were again modified for this project to collect statistics on other wireless devices that were communicating with ours. The changes made for this project's requirements were unrelated to the DSG changes.

The modified drivers were successfully tested against five different PCMCIA 802.11b network interface cards, from two different manufacturers. In all cases they worked without any discernable change in driver functionality or speed, save the features added by the modifications. The final driver version was V0.12_2b1_1. A clean version was also developed – one with the DSG modifications taken out, but the modifications made by this project remaining. This was tested on the Mandrake 8.2 operating system, were it worked successfully. The changes made are not significant enough to affect the portability of the original drivers. At a higher level, however, the developed LID protocol assumes the DSG modifications are installed and working, and these have not been tested outside of the RedHat 7.3 operating system.

5.2.1 the five PCMCIA devices tested against

The five devices the driver changes were tested with were:

- Orinoco:
  - bronze, silver and gold cards
  - Avaya "World Card", gold
- Compaq:
  - WL 100, 11Mbps Wireless LAN

5.2.2 Changes made by this project to the driver

The changes made were:

- Addition of extra data structures to the driver's initialisation routine.
- Addition of a data collection function to the statistics routine.
- Addition of an extra ioctl handling function to the driver.

5.2.2.1 Addition of an extra data structure

Each element in the added data structures corresponded to an arrived frame that data was collected on. The number of spaces in the data structures was bounded according to a rule set for a similar process in the driver. This was to ensure that any changes made would not slow the driver down.

The extra data structures held MAC addresses and the following data:

- Signal strength
- Noise

- Quality
- Whether data has been returned to client before

5.2.2.2 Addition of a data collecting routine

The existing wireless driver has functions defined for collecting similar statistics to the ones that are required for the LID protocol. All that this functionality required was the addition of an extra function, and function call where those statistics are collected. The data structures that information about received frames is collected into are modelled as a circular list. Once the list is full the index will wrap back around to the start of the list.

5.2.2.3 Addition of an extra ioctl call handler

'ioctls' are Linux / Unix system calls for general device handling. The wireless driver has a generalised handler defined that accepts an ioctl call and in turn calls a function to respond to it. The function called is decided by a flag value passed into the ioctl by the user level program calling it. A custom function call has been added using one of the un-assigned flag values from the driver – 'SIOCSIWRANGE'.

The called function locks the data structure against modification by other sections of the driver, copies the relevant data, and then unlocks the data structure. The copy made is sent to the calling process using the Linux "copy_to_user" system call.

5.3 Implementation of the LID protocol

The requirements for the LID protocol have been specified in the design chapter. Some key structures and algorithms will be shown, and then the implementation of the functional modules specified in the design chapter will be discussed.

First, a description of the structure for the network view, and the procedure for the 'receive and beacon' functionality will be made, before a brief discussion on shutdown of the LID protocol.

5.3.1 Network graph modelling: node and link structure

A network graph modelled at a node is logically a tree structure, with the device that holds the graph modelling itself as the root node. Each node 'owns' links that reference other nodes on the same tree. As a network graph will be constantly growing and shrinking, a linked list of node objects has been chosen to represent it. Thus, there are two 'views' of a network graph: how it is seen logically, and how it has actually been implemented in code.

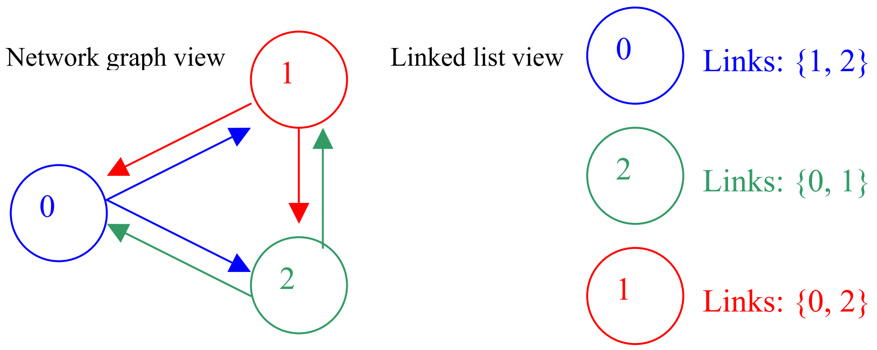


Figure 13, modelling network as a graph, modelling as a linked list

The network graph is implemented as a linked list data structure, where each node has a reference to the previous and next node on the list. Separate references are kept to the head, and current tail of the list for searching purposes. The reference to the head is fixed, while the current tail value can change as the network graph is changed.

The first node on the graph is fixed in a network graph: it is an enhanced version of a node structure that models the device that the network view is held on. This root holds a normal node structure and some extra fields.

Each node holds a linked list of 'link' data structures. These model a unidirectional connection between two nodes on the network graph. This linked list is also dynamic, and has no maximum size[15] in the network graph.

[15] That said, a single node cannot describe more than 63 links in a beacon. This was decided by space constraints, and is considered a to be a reasonable bound for information propagation. See the section on beaconing in chapter four for more detail on the space constraints of beacons.

Finally, a data type has been defined to hold the linked list, and a figure for statistics together. This comprises a history of the number of nodes described over the last number of iterations.

5.3.2 An aside on sequence numbers and node crash failures

Sequence numbers are started from zero. If a node fails and is restarted, it will have lost all of its network graph information. Upon restart, it will only know of its own existence until it starts to receive beacons again. When this node starts to receive beacons, it will compare its sequence number to the sequence number that it receives in the description of itself from the beacons of other nodes. If this value is greater than its current sequence number (allowing for wraparound), it will adopt the larger value. It will also take in the other beacon contents as usual.

This way, other nodes will ignore the beacons that a crashed and restarted node is sending until it has updated to its pre-crash sequence number - as the sequence number included will be too small compared to their record for that node. After it has updated its sequence number its beacons will be treated as valid and the process will continue as normal. This way, from another nodes point of view, a device that crashes and restarts will be viewed as having temporally been partitioned.

5.3.4 Beaconing loop

We will describe the beaconing and receiving loop here.

Upon start-up, the LID process will initialise the devices and structures associated with the protocol. If any of these fail (for example if the network device cannot be found), then the program will issue the relevant error message and exit. If the GPS device cannot be accessed, then GPS use will just be set to false and the LID protocol will continue without GPS.

Once start-up has been successfully navigated the beaconing loop will be started. This loop will run until the program is shut down. A `SIGINT` or a `SIGTERM` signal sent to the LID protocol will call the shutdown sequence. The beaconing loop executes the following actions:

- Calculate local time: t_1.
- Run the receive function – get all received beacons.
- Generate statistics from the interface about nodes that are transmitting and who are also on the network graph.
- Prune the received tree.
- Create and send a beacon.
- Calculate the number of nodes in this iteration of the network graph and add it to a history.
- Calculate local time: t_2.
- The difference in times ($t_2 - t_1$) is the execution time of the 'receive and beacon' function.
- Sleep for a specified time minus the execution time.
- Repeat.

5.3.5 Cleaning up – reactions to shutdown

As has been mentioned previously, the LID protocol is shutdown by a `SIGINT` or a `SIGTERM` signal sent to the process by the user or by the kernel initiating a shutdown. As implementation is to be in the C language, care will have to be taken to prevent memory leaks. With this in mind, a shutdown sequence has been defined that cleans up the memory allocated by the program, and closes the connection to the wireless card, etc. before exiting with a success value being returned.

5.3.6 Modules of the LID protocol

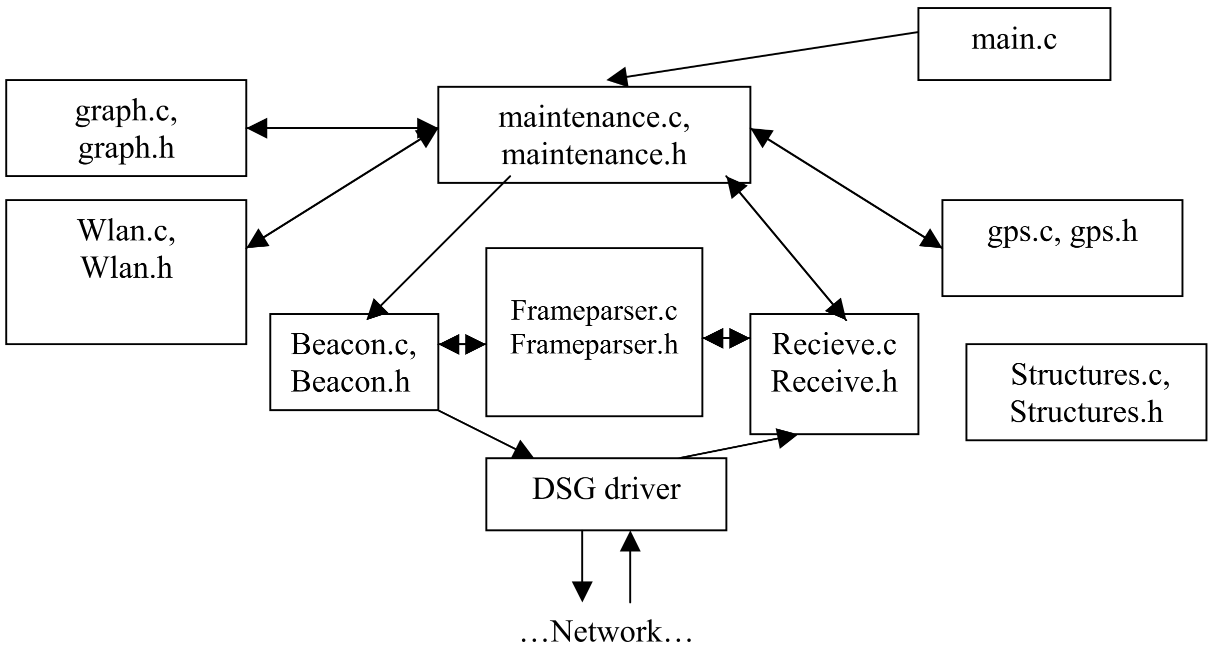


Figure 14, functional sections of the LID protocol

The sections, and their functional remit will be explained now. After this, some key data structures and algorithms used will be shown.

structures.c, structures.ch
This module is concerned with defining structures for the modelling of a network graph, and some common functions for editing it.

graph.c, graph.h
This module is used to model and maintain the network graph structure. Features include adding and removing nodes and links.

frameparser.c frameparser.h

This module is used for marshalling and un-marshalling network graphs to and from beacons.

gps.c, gps.h

The requirements section of chapter 4 stated that one desirable feature of the LID protocol was that it operated with a minimal threading model. In the case of GPS data gathering this requirement cannot be met. The GPS devices that are being used[16] are connected to the serial port of the development machines, and the GPS device updates GPS information by sending specially delimited sentences to that serial port in order to be read by a willing application. In order to read these sentences and parse them into meaningful information, the serial port reader used would have to block waiting on the port – waiting for GPS updates to arrive. This is not a useful solution in a sparse threaded model with a timed beacon requirement. To this end, the GPS reader has been implemented in a thread of its own, which has a 'sleep and check for updates' run model. The main LID application can just poll this GPS reader for its last update as required.

wlan.c, wlan.h

This module is concerned with accessing the network interface to poll for statistical information, including the range of the wireless interface. Upon initialisation, a socket is opened to the wireless driver, which is then used to poll for

- The interfaces MAC address.
- The range of the interface (a value all interfaces maintain).
- Periodic information about which other wireless devices a device has been in contact with.

[16] The GPS device used was the "Magellan 310" handheld GPS nodes used in the wireless tests of chapter 3.

transport.c, transport.h

The transport module acts as a wrapper to the DSG driver functions for sending and receiving at the MAC layer.

beacon.c, beacon.h

The beacon module will build and send a beacon on request. It is called periodically from the maintenance module.

receive.c, receive.h

This function set polls the DSG drivers in a non-blocking fashion. Returned frames are checked for validity, then parsed (using the frameparser.c functionality) and added to the network graph (using the graph.c functionality).

maintenance.c, maintenance.h

This function set is the controller of all the other modules. It initialises the various devices used by the protocol, and starts the beaconing process. It also implements the network graph pruning functionality.

One stated non-functional requirement was that the LID protocol operate with a minimal threading model. A nodes network graph can be accessed by both the beaconing and receiving functionality. If sections with possible concurrent access problems were limited to a single thread model¸ the possibility of concurrency errors / dirty reads between them can be removed. To this end, these functions are called one after the other in this section of code.

The execution of the actions in the beaconing loop are timed, and the time taken to execute an iteration of receive and beacon is taken away from the beaconing rate value to ensure that the loop then sleeps for the correct time.

main.c

This file is initialises the maintenance start up function when the program is started.

Comments

The functional layout for the LID protocol has been shown. The requirements have been met were possible, and reasons have been given as to why when they have not been.

5.4 Implementation of the PA protocol

The design chapter stated that the partition anticipator was to be divided into two sections: A general expert system framework, and a PA specific controller that uses it. The framework and controller will be described now.

5.4.1 Implementation of an expert system in C

The previous chapter introduced an expert system framework as a state machine made up of specialised state objects. In order to build this, a state machine has been designed, implementing structures for creating, holding and iterating over states. Input and state in the system will briefly be discussed before introducing the modules that maintain this expert system.

5.4.2 Nature of input into the system

A rule in a state decides if the machine should move into another state. A rule is modelled as a simple comparison function that takes in two attributes and produces one output. It is not desirable to limit inputs to being of one data type only. To this end, an attribute has been defined as a C union structure.

An attribute date structure

```
union attribute{
 int intval;
 char chVal[ATTRIB_MAXVAL];
 double dbVal;
};
```

By modeling attributes in this fashion, the C language will allow that an attribute can be one of several different data types, and as such the state machine is not limited to inferring knowledge using one input type alone.

5.4.3 Modelling of a state in the state machine

The expert system knowledge base used for this project is modelled on a normal state machine. As a result of this, there is a requirement to model a state tree that can be iterated over. A model of a state has been implemented as a C struct data type with the following contents:

A state data structure

```
struct state{
  int (*fn)(union attr *test, union attr *condition);
  union attr * condition;
  unsigned int state;

  struct state * true;
  struct state * false;
};
```

Each state holds a C pointer to a comparison function, which will return a TRUE or FALSE value. The inputs to this function are the input into the system, and an attribute value that is defined as part of a state struct. Based on the output, one of two possible pointers to the next state is chosen. If the next state chosen is NULL – i.e. it does not point to another state, then the machine has finished and the value stored in the state field is returned. This value is a flag denoting the final state of the machine.

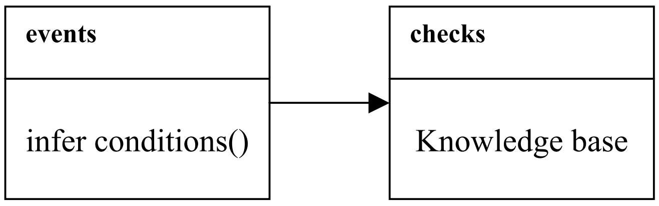


Figure 15, modules for the state machine

events

This module holds functionality for iterating through a defined knowledge base, and finding the correct knowledge base to test against. A request for analysis is passed in as an attribute value, along with a flag denoting the test to perform. The requested test will be run, and a finished state value returned to the caller. If there is an error in execution, or the caller passes in an invalid flag value, then the relevant error message will be returned.

checks

This module holds functions for knowledge base editing and holding, and attribute creation and comparison.

5.4.4 Conclusion of expert system

The state machine and a procedure for using it have been defined. The partition anticipator will now be defined. For implementation, a simple version has been created.

5.4.5 Implementing the partition anticipator

A simple partition anticipator has been implemented in order to evaluate its use. This partition anticipator will build a simple rule set, take in some data, and then evaluate it. Its purpose will be to check signal strength values and report the quality range that these values are in. This simple anticipators step-by-step functionality will be described now, with a process description and some pseudo code.

Initialise the knowledge base

Define some rules and the links between them. The signal strength rules for this PA are modelled as a 5-state machine:

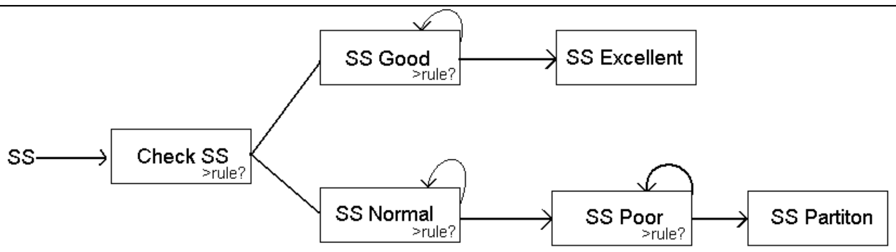


Figure 16, a simple state machine

An inputted signal strength value will be classified as being one of excellent, good, normal, poor, or partition. Each time a value is to be tested, it will be inputted and a resulting state returned. This value will be returned to the PA.

Get some data from the network view and test it

For this simple test, the PA will iterate through the list of nodes on a network graph, and for each node, examine the link quality values stored there.

```
For each node:
        For each link on a node:
        If the 'time last modified' on a link is too long ago (according to a threshold value)
              Then issue a warning about the lack of communication
       Otherwise
              Check the signal strength value using the expert system
              Report on the returned value
       End if/else statement
      End link iteration
End node iteration
```

PA results

This PA version will report the general quality of each nodes links. For this version, the focus of attention is on the PA's ability to warn that a link is entering the partition range before a node stops receiving on that link. Once no new link data is being received (indicating that a partition has occurred), then the PA should note that a link is too old and issue a report about it.

5.5 Conclusion

The implementations for a LID protocol and a partition anticipator have been described. The next chapter will evaluate this implementation and approach.

Chapter 6

Evaluation

6.1 Introduction

This chapter serves to evaluate the design and implementation of our LID protocol and PA. There will also be a discussion of the some of the possible future developments for this project.

6.2 Evaluation of the location information dissemination protocol

6.2.1 Overview

The LID protocol is capable of learning about the network that it is on. It runs upon nodes that are part of a mobile network, and those nodes can learn about each other through the sending and receiving of beacons. Each node running the LID protocol has a 'view' of the network as that node has seen it. The LID protocol serves as a memory function for a node – remembering the other nodes that this node has seen up to a threshold value of hops from this node. After this threshold, information gathered is discarded as no longer viable. Information is also time stamped, and a heuristic value is used to decide when it is out of date.

The LID protocol is similar in many aspects to a proactive routing protocol, with the difference that the 'routes' calculated are to be used for node positioning and network quality measurements rather than route table calculation – although the facility for route selection is a by-product of the LID protocol.

6.2.2 Hardware dependence

The LID protocol is hardware dependant in that it relies on both a serial port interface to a GPS device and also on the wireless driver developed. The LID protocol assumes a wireless driver that can be connected to, and functions that it can call via this connection.

The Linux operating system is especially effective at indirection, however. It is conceivable that any device dependant functionality could be replicated by indirection at another layer of Linux.

6.2.3 Location determination and motion tracking

6.2.3.1 GPS use

The LID protocol can run with GPS enabled or not. GPS has been chosen as an optional feature as it was considered worthwhile to leave open the possibility of LID protocol use in a non-GPS situation (e.g. indoors). If the LID protocol is set to use GPS but there is a failure on a node in accessing its GPS device, then the GPS option is disabled and the LID protocol will continue without it.

GPS is enabled or disabled by a node, for that node. This means that the nodes on a network can include GPS information with their beacons, or not, on a node-by-node basis. If a node includes GPS data in its beacons, then this information will be stored and propagated by other nodes in their network views, even if those nodes are not GPS enabled.

6.2.3.2 Motion tracking

The LID protocol disseminates information around the network in a controlled way. Nodes learn about each other and are able to incorporate a notion of time with this data. The LID protocol is used to gather information used as input for the PA. One avenue still to be explored is the use of the LID protocol for location and motion tracking.

As mentioned above, the LID protocol can function in a GPS or non-GPS usage model. One project [4] that was explored in the chapter 3 (the state of the art chapter) looked at the feasibility of tracking a user based on signal strength alone. Another [17] used a concept of relative and absolute location determination based on the placement of small radio nodes in several places in a building, and on users.

There are two other projects that also explore location determination [5,18] that were not documented in chapter 3 due to space constraints.

The projects studied look at location determination based on signal strength in an indoor environment. Their notions of location determination use fixed nodes (be they access points for 802.11 projects or wall mounted SpotOn devices) as reference points to infer distance from. Triangulation type techniques are generally used to determine location.

It would be worthwhile to investigate the viability of using the LID protocol for location determination in the non-GPS or partial GPS-usage (some nodes are GPS enabled, some are not) cases.

The former case would result in a mobile network where nodes can position themselves relative to each other using 2D or 3D models. The latter would result in a mobile network where some nodes locations can be known using absolute GPS co-ordinates, while some other nodes locations can only be inferred relative to each other as before or to a GPS enabled node.

These possibilities are interesting as they allow for reducing dependence on GPS in a system, and enabling use in non-GPS environments. There are several models of use for mobile networks where the assurance of proximity to an absolute location is enough to make some decision at a higher layer.

6.2.4 Warning beacons

Chapter 5 (the implementation chapter) discussed the functionality in place to ensure memory cleanup when a `SIGINT` or a `SIGTERM` signal is sent to the LID process by a user or by the kernel.

One interesting extension to this shutdown sequence that was not implemented due to time constraints was the sending of a final beacon by a device upon a LID protocol shutdown. In this beacon, the LID protocol could set all of the links to the deactivating node as being of severe partition quality, thus alerting all receiving nodes that the deactivating node is no longer on the network. This would have the nice feature that any clean operating system shutdown, where the kernel was able to send a `SIGTERM` signal to the LID protocol, would result in a mobile node issuing this warning before it was deactivated. This would be especially useful to a mobile device in the case of shutdown due to battery drainage.

Comments

Most of the requirements for the LID protocol have been met. The developed protocol is able to learn about other nodes in the network, and present this information to the PA. Network information is time stamped, and a heuristic value is used to decide if information is too old. Beacons sent are limited in range to receiving neighbours only, and the information held in a network graph is limited by a threshold value that is self adjusting, depending on the network characteristics.

The LID protocol has been tested in a two-node network. The threshold values for deciding the viability of link age and the initial L_{max} are based on heuristics. Future work on the LID protocol should take this into account, and test the protocol on a larger scale.

6.3 Evaluation of the partition anticipator

6.3.1 Overview

The developed PA iterates over the nodes and links in a network graph held on a node and issues reports about the last received signal strength of those links. The comparison is coarsely grained, classifying a signal strength value into one of five states.

The PA does not act on the advice supplied. If, in a three-node network, when two nodes find themselves partitioned from each other, but both are still able to communicate to the third, the partition is not as severe as it could have been. Communication is still possible through the third node. The PA is not concerned with this, however, as it is up to a group management protocol or partition handler to react to the classifications made about the network.

6.3.2 Testing the PA

Tests were run to determine if the PA could anticipate partitions successfully. Two laptop computers with 802.11b wireless interfaces and the LID/PA software were placed in ad-hoc mode[17].

The LID protocol was started on both laptops, and discovery of each other took place. One laptop was then carried out of range, and the partition anticipator results noted. This test was carried out three times indoors and once outdoors. The GPS functionality was not used in the outdoor test due to accuracy problems with the GPS nodes available[18]. In the tests carried out, the partition anticipator reported signal strength decreasing until a partition warning was issued, after which a partition occurred. Once a partition occurred, the PA issued a warning that the links in

[17] This was to prevent interference from the 802.11b driver switching between access points during the test. The 'mode' (access point, ad-hoc, or hybrid based) of the wireless does not affect toe operation of the LDI protocol.

[18] The GPS nodes used for this project were unreliable in terms of co-ordinates returned. For example, when the nodes were held next to each other at different times, they reported distances between 10 and 70 meters apart.

question had not been updated since before a threshold time. The moving node would then move back towards the stationary one, and the nodes would rediscover each other and the signal strength would improve.

6.3.3 Current version of the PA

The tests performed show the viability of the LID/PA protocols for partition anticipation. However, the version of the PA used did a simple analysis based on current signal strength only. The knowledge base created did not permit for complex analysis of the network graphs that were built.

6.3.4 Use of GPS data

Given the differences in distance ability between different brands of 802.11b interface recorded by the tests of chapter 3, the potential for the use of distance as a metric in partition anticipation seems limited. Where wireless interfaces that transmit at the same power can have such radical differences in ability to communicate over distance, the choice for a PA is either to learn which type of interface it is using, or to assume the worst case. Neither of these is an attractive option.

6.3.5 Extension of the partition anticipator

There are several possibilities for partition anticipation that are worth proposing under the auspices of future development. These will be discussed now.

6.3.5.1 Optimistic and pessimistic use

Killijian et al [7] discussed the notion of optimistic and pessimistic modes for a partition anticipator. This would mean that a PA would learn from its positive and negative misclassifications.

One approach to this was the idea that a PA system could learn this sort of behaviour by taking feedback from the actual results of a prediction, and changing the knowledge used for inference – increasing or decreasing the threshold values, to take the signal strength classifier as an example. This sort of system is termed a *classifier* *system*, and it is implemented as a combination of genetic algorithms[19] and a normal expert system.

It remains to be seen if this would be a successful property for a PA to have, however. Learning about the network this way could be too slow, as conclusive feedback into the system could only really come in the form of anticipated or unanticipated partitions.

Using feedback in this fashion to alter the knowledge base is also based on the assumption that a network will not fluctuate rapidly, resulting in the system flicking between optimistic and pessimistic when there is no need. For example, a PA might enter pessimistic mode if it was cut off from its neighbours due to a rash of unanticipated partitions, only to have the cause of those partitions be removed just as quickly, leaving it to slowly learn to be optimistic in a strong network environment. Certainly, an approach such as this would have to be cautious in its decisions, as to move too quickly in either direction would be to invite misclassification.

An alternative approach might be for a PA to use the values for average number of nodes connected to, average noise, etc. as a general quality figure that an optimistic or pessimistic mode can be inferred from. This solution would be more stable, as average figures kept over a time are less likely to be affected by short term deviations.

6.3.5.2 A partition anticipators information about the device it is running on

Currently, the PA takes in information from the LID protocol only. The wireless tests from chapter 3 indicated that, among other things, the speed of sending frames and the size of those frames might be factors contributing to the probability of a message not arriving at its destination due to loss or message corruption. Regularly updated average values for the speed and size of frames could be included in the knowledge base for the partition anticipator, and these could be used to influence the PA towards optimistic or pessimistic use similar to that of the above section. The LID protocol could be extended to collect these values without much altering.

Some other general pieces of information that could prove useful to a PA are:

[19] A genetic algorithm is one that mimics natural evolutionary processes in order to solve computational problems.

- The transmitting power of the interface.
- The estimated range of the interface.
- The type of transmitter / receiver / aerial being used.

The first two could easily be gotten from the wireless interface, and the aerial 'type' could be a general inference, decided on the back of a higher than normal transmitting power and receiving gain[20]. These values are interesting as they can describe a device's power in terms of transmitting and receiving, and hence make more by way of assurance about the quality of the network at this node.

6.4 A grammar for knowledge

The developed PA has the ability to build knowledge bases that can be used to classify network quality. It might prove worthwhile if a structured grammar was developed that could describe these rules in a format that the PA could understand. Rules could be modelled outside the PA program and could be loaded from an external source at program start, or dynamically during runtime. This addition would act to further separate the PA from the current set of heuristics that it was using.

6.5 Limitations of a partition anticipator

The Y-axis rotation test discussed in chapter 3 described an effect that occurred when a person held a wireless device, A, blocking the line of sight between it and another device, B, that was 150 meters away. At B, all transmissions from A were suddenly blocked, and transmission was not possible again until the blocking person began to move around again. This effect is in keeping with comments made during the RADAR project [4] (discussed in chapter 2), which observed a similar effect, albeit with not such drastic effects. The loss in transmission during the 802.11b tests was

[20] A powerful receiver can increase the signal strength of a received transmission by amplifying the signal. This is referred to as gain.

immediate, and there was none of the normal fading in signal strength observed during the other tests.

These observations highlight the limits of use for a partition anticipator, in that they proved that a partition could occur without warning, from something as small in scale as a user turning around. The PA developed for this project can notice partitions like these by observing that a link has not been updated since a certain time – a threshold time value beyond which a links information is considered stale.

6.6 Comments

The PA is able to infer the possibility of a partition from a low quality connection between nodes. The rules that it uses for this are somewhat crude, and would benefit greatly from being refined. The smaller the granularity that behaviour can be classified to, the more accurate the PA will be.

6.7 General comments

6.7.1 Inter process communication (IPC)

6.7.1.1 LID protocol

An open issue with this project is of access to the data structures developed from other programs. Two options considered for this type of access were shared memory, and a more explicit request-response based IPC call. The use of system interrupt calls for performing requests was also an option, but this was rejected out of hand as it would most likely interfere with the beaconing procedure at an inopportune time.

The LID protocol models a dynamic data structure, the structure and size of which could radically change from one moment to the next. The ability to set values in the network graph from a higher-level process might well be a desirable feature. One example might be a higher-level protocol removing a node from the network graph, or changing the value for L_{max}.

Due to the dynamic size of the information that the LID protocol holds, shared memory is an unsuitable paradigm for out of process access, in spite of its efficiency implications[21]. Access to the network graph structure from an out of LID protocol process needs to be tightly controlled, as such access is at a high risk for dirty reads and other concurrency problems.

This left the IPC request-response option. This was not investigated due to time constraints, but it would not be difficult to imagine the development of an interface for requests through IPC in the LID protocol. Timed asynchronous requests could be passed to the LID protocol, which would process these requests and return replies at an opportune time in its beaconing cycle. This option would be less efficient than the shared memory model, but would be safer in terms of concurrency. One drawback is the asynchronous nature of the requests. A higher layer would have to issue a request and then wait an undefined amount of time for the response. Implementing a synchronous IPC request-reply model would again result in the possible interruption of the LID protocol at an inopportune time.

The current version of the LID protocol solves this problem by not providing a feature for it. There is no procedure for out of LID protocol access to the data structures therein.

6.7.1.2 The Partition anticipator

The PA protocol currently runs inside the LID protocol process space.

One desired function of the partition anticipator was that it be able to report to another process. It was mentioned in the design chapter that IPC or system interrupts could be used for this process.

The existing PA protocol works on a request-response principle. A potentially large set of checks is run and the results for each check returned. Partition anticipation itself seems better suited for an event driven model of use, where different processes, or different parts of the same process, could set specific conditions to watch for and give a call back reference for when an event does occur. When observed, these

[21] In Linux, shared memory works by sharing a segment of computer memory between several processes. This is highly efficient for IPC as both processes are working directly with the values involved instead of passing variables to each other by value.

conditions could result in an event being generated in the process that requested the monitoring. This seems to be a better usage model than calling an interrupt in another process, or writing an IPC request message which the other process would then have to block on / periodically check for anyway.

6.7.2 The LID protocol and the future of 802.11b

The LID protocol developed during this project uses the 802.11b standard as its wireless communication protocol. The LID protocol works on the MAC layer. The different standards of 802.11 are physical layer standards, so in theory the LID protocol, or a modified version of it, could run on any 802.11 compliant standard that was radio frequency based. An alternative to the existing driver extensions that make the LID protocol possible would have to be sought.

Certainly, it would be interesting to look at the feasibility of LID and PA functionality in a higher bandwidth standard such as 802.11a. The 802.11g working group is currently considering an extension of 802.11b that will extend the data rate to 22 Mbps. This extension will be backward compatible, and again provided the same driver functionality will be able to support a LID or LID type protocol.

6.8 Chapter conclusion

This chapter has looked at the implementation of the LID protocol and the partition anticipator. Some aspects of these two programs were highlighted, and design alternatives were discussed. The chapter also identified some limits of partition anticipation, and discussed possible ways to increase the effectiveness of a partition anticipator.

Chapter 7

Conclusion

Objectives revisited

One objective for this project was the development of understanding about the chosen wireless technology, and the identification of metrics that could be used to identify network quality. Another was the building of a partition anticipator and a network information service that could supply the PA with fresh information.

What has been achieved in this project

This dissertation has broached the design and implementation of a partition anticipator and a location information dissemination protocol. Both were implemented in the C programming language and run on the Linux operating system.

Experiments have been run on 802.11b that provided a rich set of data. The analysis of this data has provided enough knowledge about the wireless environment to facilitate the identification of metrics for the description of a wireless network, and specific and general case rules for network quality classification. The location information dissemination protocol has been designed to monitor and record the network in terms of these metrics. The partition anticipator has been designed as being made up of two parts: one to model the rules for network quality and classify input into a value for network quality, and the other to provide input to these rules and report the classifications generated.

The PA and LID protocol have been implemented, and have been tested successfully in accordance with the goals identified in the chapter 4 (the design chapter). As a corollary, an extension has been built for the PCMCIA wireless driver for monitoring the local network. Also, the LID protocol provides a feature for maintaining an OSPF routing table on each node.

Summation

This dissertation dealt with wireless networks and the desire to understand them better. A chosen wireless technology was explored, and a means for identifying and classifying the behaviour of the devices that use it was developed. This project has identified key issues underlying wireless communication, and proposed several interesting avenues of research that remain to be explored.

Bibliography

[1] G.C. Roman, Q. Huang, A. Hazemi, “Consistent Group Membership in Ad Hoc Networks” Washington University, St Louis

[2] R. Prakash and R. Baldoni, “Architecture for group communication in Mobile Systems” presented at Symposium on Reliable Distributed Systems, West-Lafayette (IN), USA 1998

[3] T. Goff, N.B. Abu-Ghazaleh, D.S. Phatak, R. Kahvecioglu “Preeemptive routing an Ad Hoc Networks” University of Maryland, Binghamton University

[4] P. Bahl, V. N. Padmanabhan “RADAR: an In-building RF-based User Location and Tracking System”. Published with some revisions as “User Location in an in-building Radio Network”, Technical Report, MSR-TR-99-12.

[5] M.A. Youssef, A.Agrawala, et al “A probabilistic Clustering-Based Indoor Location Determination System”, Dept of Computer Science, University of Maryland. UMIACS-TR-2002-30

[6] R. J. Punnoose, P. V. Nikitin, et al “Optimising Wireless Network Protocols using Real-Time Predictive Propegation Modelling”, Dept of Electrical and Compute Engineering, CMU, PA

[7] Marc-Olivier Killijian, Raymond Cunningham, René Meier, Laurent Mazare, and Vinny Cahill. “Towards Group Communication for Mobile Participants”. In Proceedings of ACM Workshop on Principles of Mobile Computing (POMC 2001).

[8] G.C. Roman, Q. Huang, A. Hazemi, “On Maintaining Group Membership Data in Ad Hoc Networks” Washington University, St Louis

[9] Y-B Ko, N. H. Vaidya “Location Aided Routing (LAR) in Mobile Ad Hoc Networks”. Dept of Computer Science. Texas A&M University, TX 77843-3112

[10] C-K Toh, “Associativity-Based Routing for Ad-Hoc Mobile Networks” University of Cambridge, Cambridge, UK

[11] L. Blazevic, L. Buttyan, S. Capkun, et al. “Self-Organization in Mobile Ad-Hoc Networks: the Approach of Terminodes”. Institute for computer Communications and Applications (ICA). Swiss Federal Institute of Technology, Lausanne, Switzerland.

[12] M.I. Silventoinen, T. Rantalainen “Mobile station emergency locating in GSM” In IEEE International Conference on Personal Wireless Communications, 1996

[13] Jerome H. Saltzer, David P. Reed, David D. Clark “End-To-End Arguments In System Design” (1984), ACM Transactions on Computer Systems

[14] “GeoDoom: a location aware game” A Tisserant, Trinity College Computer Science, DSG 2002.

[15] “Wireless Extensions for Linux”, Jean Tourrilhes, Hewlett Packard. January 97

[16] “Sending and receiving data without IP” Distributed Systems Group, Trinity College Computer Science, DSG 2002.

[17] “Design and Calibration of the SpotON Ad-Hoc Location Sensing System” J. Hightower, C. Vakili, G. Boriello, R. Want. University of Washington, Computer Science and Engineering.

[18] “Location Detection in a Wireless 802.11b Network Environment” E. Bhasker. An abstract for location detection software developed, CalTech,UCSD, Ca.

[19] “Wireless LANs: Implementing High Performance IEEE 802.11 Networks” chapter 1, J. Geier. Second Edition. SAMS publishing.

[20] “Wireless Communication, The Interactive Multimedia CD-ROM” section on radio propagation, Jean-Paul M. G. Linmartz
Baltzer Science Publishers, P.O.Box 37208, 1030 AE Amsterdam,

[21] “Wireless Communications: Principles and Practice” T. Rappaport, Prentice Hall publishers

[22] “What is a Wireless LAN?” White paper 2001, Proxim Inc.

[23] “IEEE Standard for Wireless LAN Medium Access (MAC) and Physical Layer (PHY) Specifications” ISO/IEC 8802-11:1999. IEEE, 1999

[24] “…Roaming with WaveLAN/IEEE 802.11” Lucent Technical Bulletin 021/A, Dec 1998

[25] “Data Collection Application for 802.11 WLANs” Gregor Gärtner , Distributed Systems Group, Trinity College Dublin. v0.99 May 2002

[26] “ORiNOCO™ World PC Card” Agere Systems Inc., 2001. product specification for Orinoco Silver PCMCIA wireless interface.

[27] Compaq WL100 11Mbps Wireless LAN PC Card Specifications.