Network-Layer Reservation TDM for Ad-Hoc 802.11 Networks

Submitted in fulfilment
of the requirements of the degree
Master of Science
of Rhodes University

Kevin Craig Duff

Supervisors: Prof. Alfredo Terzoli, Prof. Peter Clayton

February 26, 2008

Abstract

Ad-Hoc mesh networks offer great promise. Low-cost ad-hoc mesh networks can be built using popular IEEE 802.11 equipment, but such networks are unable to guarantee each node a fair share of bandwidth. Furthermore, hidden node problems cause collisions which can cripple the throughput of a network.

This research proposes a novel mechanism which is able to overcome hidden node problems and provide fair bandwidth sharing among nodes on ad-hoc 802.11 networks, and can be implemented on existing network devices

The scheme uses TDM (time division multiplexing) with slot reservation. A distributed beacon packet latency measurement mechanism is used to achieve node synchronisation. The distributed nature of the mechanism makes it applicable to ad-hoc 802.11 networks, which can either grow or fragment dynamically.

Keywords: 802.11, ad-hoc, access method, fairness, hidden node, multi-hop, TDM

Acknowledgements

I dedicate this work to my parents, Ron and Rose. Their being tasked with me has been thankless at times, I'm sure, so here's a thank-you, folks.

Alfredo Terzoli, my "day-to-day" supervisor, thank you for your time, your support, your infinite patience, but most of all for your faith in me. I'd never have finished, were it not for your encouragement. Peter Clayton, my "executive" supervisor, thank you for your guidance and support.

The Melbourne Wireless Community, authors of Frottle. From you I borrowed ideas, code fragments, and a head-start on my solution. And thanks to the other researchers, whose work I cite herein.

Thanks also to the staff and students of the Computer Science department at Rhodes University. Pat Terry, George Wells, Caro Watkins - thank you.

Robyn, my girl, for being there for me during the highs and lows. Mike Theron, my moral supporter, part-time conscience and stress reliever.

Finally thank you to my sponsors, the Telkom Centre of Excellence at Rhodes University, funded by Telkom SA, Business Connexion, Verso Technologies, THRIP, Stortech, Tellabs and the National Research Foundation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogroves,
And the mome raths outgrabe.

Lewis Carroll, The Jabberwocky

1.1 Introduction and Motivation

Recent years have seen a proliferation of low-cost IEEE 802.11-based wireless networking devices. Almost all laptop computers sold today are equipped with an 802.11 interface. Cellular telephones are being fitted with secondary 802.11 interfaces to extend network coverage, through technologies such as UMA [20]. 802.11 is freely available, operates in an unlicensed bandwidth, is widely deployed, and has reached a level of maturity such that interoperability is good, largely as a result of the Wi-Fi [21] standards.

These devices support an ad-hoc mode, which enables peer-to-peer communication between wireless devices without the need for any central access point. Appropriate routing protocols can be utilised to enable multi-hop networks.

802.11 in ad-hoc mode uses CSMA/CA[1] to resolve contention for the physical medium. The use of CSMA/CA results in two problems:

[1] CSMA/CA is described in section 2.6.3.

- CSMA/CA employs a first-come-first-served scheme. Under this scheme, some nodes can hog available bandwidth, thus depriving or even starving other nodes of bandwidth[2].
- In networks that are not fully connected, 802.11 suffers from Hidden Node problems. Hidden Nodes cause collisions and loss of data, and can severely reduce network performance[3].

This research aims to remedy these problems, by combining two sets of ideas:

- Research into Layer 2 protocols for ad-hoc networks has yielded MAC schemes which divide time up into slots (or slices). A node must successfully reserve a slot before it is allowed to transmit any data. These distributed algorithms schedule transmissions in advance to avoid hidden node problems and fairly share bandwidth. Because these protocols modify Layer 2, they are not compatible with existing 802.11 devices.
- Applications have been developed to overcome hidden node problems, and provide fairer bandwidth sharing, on existing 802.11 devices. They commonly employ a token-passing scheme, passing the token at the transport layer. In such applications, a single master node controls a number of slave nodes. Because a master node is required, these applications do not suit the ad-hoc paradigm and are normally used only on infrastructure[4] networks.

If the asynchronous channel provided by existing 802.11 devices could somehow be split into discrete time slices, a synchronous virtual slotted medium would become available.

A slot reservation scheme could be implemented above the virtual slotted medium so as to overcome the hidden node problem, and achieve fair bandwidth sharing.

If the mechanism could be compatible with, and applicable to, existing 802.11 devices, it could overcome two real barriers to building ad-hoc mesh networks with 802.11 equipment. Such a mechanism could facilitate many exciting new possibilities with ad-hoc networks on current equipment.

The low capital costs involved would suggest that the technology might be particularly applicable for providing telecommunications to poor or previously disadvantaged communities.

[2] See section 2.7

[3] The Hidden Node problem is described in section 2.8

[4] 802.11 infrastructure networks, requiring central access points, are discussed in section 2.10.2.

1.2 Research Objectives

The objective of this research is the design of a software mechanism which:

1. Provides each node with a fair share of available bandwidth
2. Overcomes Hidden Node problems
3. Is applicable to existing 802.11-based devices

1.3 Research Methodology

A survey of literature relating to the hidden node problem in ad-hoc networks yielded various schemes which share bandwidth fairly and overcome hidden node problems. By identifying common elements in these schemes, we were able to derive a design for implementing such schemes at Layer 3.

Each component of our design could feasibly be implemented in various ways, depending on the application for which the framework is being employed.

In order to test the feasibility of our framework, we designed a prototype application.

Subsequent research involved an iterative cycle of analysis and design, implementation and testing. A prototype software application was iteratively developed and deployed on a small network of nodes in a laboratory. A description of the development environment can be found in chapter 6. The behaviour and performance of the application was studied and the issues identified were addressed in the next iterative cycle.

The various issues which were identified and addressed during these cycles were documented, and may be of interest to future research in this area.

The performance of the application was tested for various metrics under various conditions. The results of the performance evaluation are presented graphically and the findings are discussed in chapter 8.

1.4 Evaluation

The software mechanism will be evaluated by addressing the following research questions:

- What are the benefits of the mechanism?
- What complications and issues arise?
- What limitations can be identified and quantified?
- What are the costs, in terms of latency and bandwidth, of the mechanism?
- Does the mechanism promise a useful solution to the research objectives?

1.5 Document Overview

The rest of the document is organised as follows:

Chapter 2 introduces some background material which is relevant to an understanding of the problem area, as well as to the proposed solutions.

Chapter 3 reviews previous work relating to the problem area.

Chapter 4 discusses and evaluates the requirements for a software application to facilitate the research objectives.

Chapter 5 proposes software designs to meet the requirements discussed in the previous chapter.

Chapter 6 describes a prototype Linux application which was developed during this research. The various issues which were encountered during development are reported in this chapter.

Chapter 7 documents experiments which were performed during the course of this research.

Chapter 8 seeks to evaluate the performance of the software solution.

Chapter 9 concludes the dissertation. The research objectives and questions are reviewed, future work is proposed, and the specific contributions made by this work are discussed.

Chapter 2

Background and Research Context

This chapter introduces some background material which is relevant to an understanding of the problem area, as well as to the proposed solutions.

2.1 Introduction

This chapter introduces material relevant to an understanding of the research objectives. The topics discussed are summarised below:

- Ad-hoc networks are introduced
- Time Division Multiplexing (TDM)
- Access Methods, including CSMA/CA
- The Hidden and Exposed node problems
- The fairness of 802.11 bandwidth allocation
- 802.11 in the Context of Ad-Hoc Networks

2.2 Ad-Hoc Networks

This section provides some background information regarding types and applications of ad-hoc networks.

2.2.1 Introducing Ad-Hoc Networks

Ad-hoc networks use ad-hoc routing protocols to create a logical mesh topology[48]. Each node can act as a repeater, resulting in a network that can span large distances. Because each node needs only to transmit as far as the next node, infrastructure is inexpensive and decentralised.

Ad-hoc routing protocols, such as OLSR[30] and AODV[47], are able to forward data across multiple hops to reach the destination node. Because nodes in an ad-hoc network are commonly able to communicate with more than just one neighbouring node, redundant routes become available. Ad-hoc wireless networks are thus implicitly mesh networks.

2.2.2 Mesh Networks

Mesh networks are generally very reliable, because each node is connected to several others. If a node which is routing data fails, the network is able to find an alternative route. Extra capacity can theoretically be installed by simply adding more nodes[1]. Mesh networks may be fixed or mobile.

2.2.3 Community Networks

Wireless community networks [8, 9, 14, 13, 19, 10] take advantage of the recent development of cheap, standardised 802.11 devices to build metropolitan area networks [53].

Some community networks are used to share Internet connectivity, particularly where unmetered connections (such as ADSL) are available at fixed costs.

Community networks also offer the potential for free telephony using Voice over IP (VoIP). Some community networks are used to carry video signals from security cameras.

There have also been cases of municipalities successfully deploying community networks. 802.11-equipped wireless devices can be attached to street lamps, from which they can draw their power. This is a cost-effective way to deploy a metropolitan area network.

The ad-hoc paradigm would seem to suit community networks well. The reality is that most community networks presently use 802.11 infrastructure networks with access points connected to a backbone.

[1] Because 802.11 utilises a shared channel, the addition of a single-channel 802.11 node will normally not increase aggregate throughput.

Within South Africa, community networks are being deployed in metropoles such as Gauteng[7], Durban[3] and Pretoria[12]. The CSIR also operates a research community network[2].

Under current telecommunications legislation in South Africa, the legality of community networks is questionable.

2.2.4 Military Applications

Ad-hoc networks are of great interest in military applications because the lack of fixed infrastructure means that they can be deployed quickly and arbitrarily in a battlefield situation.

2.2.5 Sensor Networks

Sensor networks are ad-hoc networks consisting of simple nodes, normally battery-operated, equipped with a wireless network interface. They have many applications, such as monitoring of machinery.

An interesting application for sensor networks is in tracking of forest or mountain fires. Sensor nodes with GPS[2] facilities could be scattered across the area of interest, to track the progress of the fire.

2.2.6 MANETs

The physical topology of a Mobile Ad-Hoc Network (MANET) is very dynamic, because individual nodes are able to move. This makes routing in a MANET challenging.

An interesting application for MANETs is vehicular networks. In such networks, moving cars would be able to communicate with one-another as well as with passing street lamps, traffic lights, etc. Such networks could be utilised for conventional data and voice applications, as well as to relay traffic information such as speed limits, congestion information, warning of accidents or road works, etc.

[2]Global Positioning System

Figure 2.1: The LinkSys WRT54G 802.11g Wireless Router

2.3 OpenWRT: An Ideal Platform for Community Networks?

OpenWRT[18] is a Linux distribution for wireless routers. The OpenWRT project commenced after LinkSys[17] made the pioneering move of releasing the source code to their WRT54G[23] 802.11g wireless router under the GNU GPL.

Developers soon began to modify and tweak the LinkSys firmware. OpenWRT is an open-source re-write of the proprietary parts of the LinkSys firmware.

OpenWRT has subsequently been ported to numerous wireless routers and access points other than the WRT54G, from a wide variety of manufacturers.

The Linksys WRT54G[23] is an 802.11g wireless router, based on the Linux operating system. The device includes a 4-port Ethernet switch, a wireless interface, 16 or 32 Mb RAM and software-upgradeable firmware. The device is favoured by community networks because it is reliable, and because the firmware can be modified to run OpenWRT. OpenWRT allows for great flexibility in customising the device.

The OpenWRT operating system includes a package manager, ipkg, which makes it reasonably easy to install additional applications.

Wireless routers similar to the WRT54 are presently available at a relatively low cost (less than ZAR 1 000,00). Combined with OpenWRT, they offer the promise of an ideal low-cost node building block for ad-hoc networks. Users could connect to the ad-hoc network via the Ethernet ports on the wireless router, or wirelessly.

Some wireless routers are even equipped with dual 802.11 interfaces, which could be configured to improve the performance of a mesh by partitioning data into different channels. By partitioning a network into two or more 802.11 channels, the number of nodes sharing each channel can be reduced thus each node experiences increased throughput.

In conclusion, OpenWRT running on low-cost hardware offers a cost effective means of building network nodes. The integrated packet manager could feasibly be used to deploy a software application such as this research aims to develop.

2.4 Signalling vs Payload Data

In any communications network, some form of control is required to set up and tear down communications channels [46]. In the telecommunications arena this control is defined by a signalling protocol. The signalling protocol defines the operations required to control a specific communications medium.

Payload data may be defined as the actual data which an application requires a network to transmit. Signalling data, on the other hand, is the additional data which the network uses to control the payload transmission.

When signalling and payload data share a common channel, this is known as in-band signalling. When separate channels are allocated for signalling and payload data, it is known as out-of-band signalling.

2.5 Time Division Multiplexing

TDM divides the capacity of a medium into time slots. Each slot is able to carry a separate data stream. Because the sender and receiver are synchronised, they are able to identify the slots and recreate the original data streams.

Because TDM can offer bandwidth guarantees to nodes, Quality of Service (QoS)[54] services like IntServ[5] or DiffServ [4] can be implemented. QoS facilitates real-time media, such as Voice over IP telephony.

2.6 Access Methods

Nodes in an ad-hoc network commonly have to share a physical medium, with the requirement that no two nodes within communication distance of each other may transmit simultaneously. Nodes may only transmit one-at-a-time, otherwise collisions occur and the transmitted messages are not received properly.

A channel access method or multiple access method allows several terminals connected to the same physical medium to transmit over it and to share its capacity. Access method protocols for a single channel medium may typically be categorised as token-based or multiple-access [26].

2.6.1 ALOHA

ALOHA[24, 25] was an early wireless computer networking protocol. ALOHA allows nodes to transmit at arbitrary times. The protocol waits for acknowledgements to all transmissions, and re-sends data which has not been acknowledged after a certain time has passed.

Because there is no mechanism to avoid collisions, they occur frequently under ALOHA and this leads to very low throughput rates. A later improvement, Slotted ALOHA[51], divided time up into discrete slots. By requiring nodes to transmit at the start of a time slot, it managed to improve throughput significantly.

2.6.2 CSMA/CD

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) is the access method utilised by wired Ethernet networks. CSMA enables multiple devices to share a common channel or medium, by enabling a node to detect when another node is transmitting. A node may not commence a transmission while another node is transmitting. However, Ethernet transmits bits so rapidly that multiple bits can be sent over a long cable before they arrive at the receiving end.

Meanwhile, a node at the receiving end can sense the medium as free and simultaneously commence a transmission, leading to a collision. CD enables the nodes to detect the collision, after which both back off for a random period before attempting a retransmission.

2.6.3 CSMA/CA

Carrier Sense Multiple Access with Collision Avoidance [35] (CSMA/CA) is the access method predominantly on 802.11 wireless Ethernet networks. A node wishing to transmit must first sense the medium, to discover whether another node is transmitting. If the medium is not busy, the node may proceed with transmission.

2.6.3.1 Inter-frame Gap

With high network utilisation, the period immediately following a busy medium has the highest possibility of collisions. CSMA/CA enforces a minimum time gap between frames from a given node. After transmitting, a node must wait until the gap has expired before attempting to transmit again.

2.6.3.2 Random Back-off Period

Once the inter-frame gap has expired, a node wishing to transmit must select a random amount of time to wait (the back-off interval) before attempting to transmit.

If the medium is still busy, the node selects a another random back-off interval which is shorter than the previous. This process is repeated until the random back-off approaches zero and the station is allowed to transmit.

The random back-off period is designed to ensure judicious channel sharing.

2.6.4 RR-ALOHA

RR-ALOHA [27] uses a slotted structure, with slots grouped into virtual frames. Slotting information can be provided by an external clock, such as GPS. It can also be implemented on asynchronous physical layers such as that provided by 802.11. Under RR-ALOHA, nodes contend to reserve an available slot. Upon success, that slot is reserved in subsequent frames and

not accessed by other terminals until it is released. The slot is known as a basic channel (BCH), and it provides terminals with a reliable broadcast channel not suffering from the hidden-node problem[3]. In addition to transmitting payload data, nodes use it to broadcast their view of the state of each slot in the frame. RR-ALOHA identifies hidden nodes and prevents hidden node problems as follows:

When a node (N1) receives information of a slot reservation by second node (N2) via the broadcast of a third (N3), but N1 does not receive N2's broadcast, N1 marks N2's slot as reserved. N1 has identified N2 as a hidden node and avoids hidden node problems by not using N2's slot. N2 becomes aware of N1 in a the same way. RR-ALOHA also includes a mechanism to overcome the exposed node problem[4], enabling parallel point-to-point transmissions, and offers an efficient broadcast service.

2.7 The Fairness of 802.11 Bandwidth Allocation

The 802.11 MAC uses contention (via CSMA/CA) to decide which node is allowed to transmit. Under such a scheme, bandwidth is allocated on a first-come-first-served basis.

CSMA/CA's random diminishing backoff (described in section 2.6.3) is designed to provide judicious channel sharing.

Gambiroza *et al* [34] provide a comprehensive study of fairness and end-to-end performance in wireless multi-hop networks. Their results show that current 802.11 protocols can result in severe unfairness, and even total flow starvation of some nodes in a network

An investigation [56] identifies several serious fairness issues with the 802.11 MAC when used for multi-hop ad-hoc networking. They conclude that the 802.11 MAC does not function well in multi-hop networks.

A performance anomaly [39] of 802.11b was noted by Heusse *et al*. CSMA/CA is designed to guarantee an equal long-term channel probability access to each host. However, hosts with low bit rates are likely to capture the channel for longer periods. Because the channel is a shared medium, hosts with higher bit rates are thus penalised.

[3] See section 2.8 for further information on the Hidden Node Problem.
[4] Section 2.9

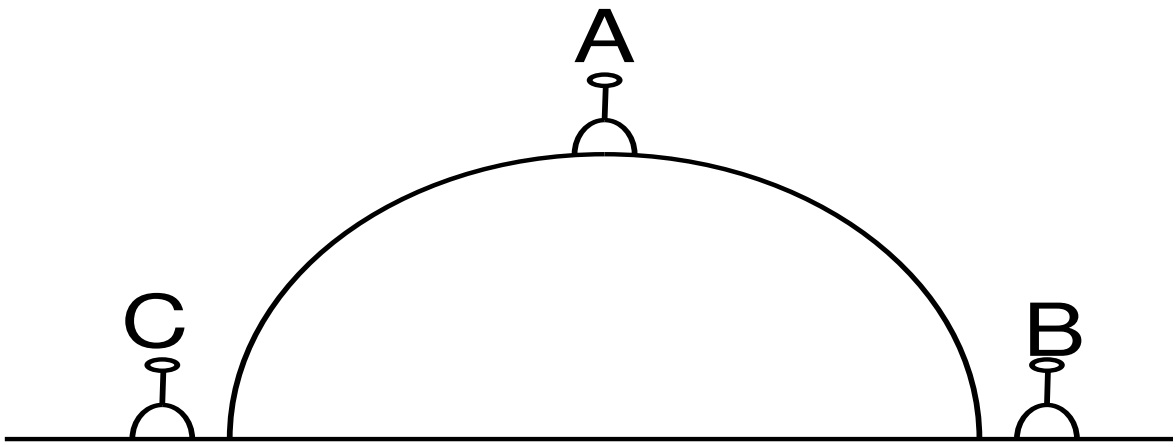


Figure 2.2: Illustrating The Hidden and Exposed Node Problems

2.8 The Hidden Node Problem

The Hidden Node (or Hidden Terminal) Problem affects wireless networks which share a common physical medium and use carrier sensing for contention resolution. The 802.11 Distributed Co-ordination Function (DCF) uses CSMA/CA, and thus suffers from hidden node problems. Under CSMA/CA a node must wait until other nodes are silent, before it is allowed to transmit.

Figure 2.2 illustrates the hidden node problem:

- A can hear both B and C, but B and C cannot hear one-another
- Because B and C cannot hear one-another, CSMA/CA cannot co-ordinate these nodes properly.
- B and C transmit simultaneously, resulting in collisions at A

A hidden node, from the perspective of the transmitter, is a node within range of the receiver but out of range of the transmitter [38]. Thus, in figure 2.2, C and B are hidden nodes to one-another.

Kahol *et al* [52] have quantified, through simulation, the detrimental effects of hidden node problems. Their findings indicate that if more than 10% of the nodes in a network are hidden, the throughput of the network drops dramatically. In a network where 30% of nodes are hidden, throughput degrades to a mere 22% of the full capacity.

A comprehensive review of point-to-point protocols which are able to overcome the hidden node problem is available in [38].

Ad-hoc networks are especially prone to hidden node problems, as section 2.10.5.2 will explain.

2.9 The Exposed Node Problem

The Exposed Node Problem prevents parallel transmission by nodes, even when such transmissions could take place without causing collisions.

Refer again to figure 2.2. Assume that node A is transmitting to B. The CSMA/CA mechanism would prevent any transmissions by node C. Although C could transmit to nodes other than A without causing collision, it is prevented from doing so by CSMA/CA.

The exposed node problem leads to reduced network efficiency.

Because CSMA/CA prohibits transmission when a carrier is sensed, the exposed node problem cannot be overcome in existing networks unless the MAC is modified to override CSMA/CA.

2.10 802.11 for Ad-Hoc Mesh Networks

2.10.1 Introduction: IEEE 802.11

The IEEE 802.11 working group [16] has defined Medium Access Control (MAC) and Physical Layer (PHY) standards for wireless connectivity within a local area. Aspects of the 802.11 MAC which are relevant to this research are discussed below.

2.10.2 Infrastructure vs Ad-Hoc : Service Sets

Commonly 802.11 networks utilise a central Access Point (AP) to provide connectivity to a wired network, such as the Internet. However, 802.11 supports networks which have no APs (ad-hoc networks) as well as those with one or more APs.

2.10.2.1 Basic Service Set and Ad Hoc Mode

A peer-to-peer wireless network which requires no AP is known in 802.11 terminology as a Basic Service Set (BSS). All stations are kept within a circular radius of approximately 300 feet so that nodes can communicate with one-another directly. BSS uses the Distributed Coordination Function (DCF), described in section 2.10.3.

No routing is required in a network where all nodes can communicate with one-another directly.

2.10.2.2 Access Points and Infrastructure Networks

More commonly, 802.11 networks utilise a single central Access Point (AP). The AP is used as a bridge or a router, to provide connectivity to a wired network. Networks utilising APs are known as Infrastructure Networks.

An AP provides authentication and association services. APs are able to temporarily silence other nodes for point-to-point transmission of time-sensitive data. APs are also able to alleviate hidden node problems, through functions like the PCF[5] and RTS/CTS[6].

2.10.2.3 Extended Service Sets

A logical collection of BSSs is called an Extended Service Set, or ESS. Multiple APs can be linked to form an ESS, enabling clients to roam between APs.

2.10.3 DCF and PCF: The 802.11 Coordination Functions

The 802.11 MAC provides for two coordination functions, which are used to resolve contention for the physical medium: The Distributed Coordination Function (DCF) and the Point Coordination Function (PCF).

The DCF is the prevalent access method of 802.11. It provides controlled access to the shared wireless medium via CSMA/CA. Because it uses CSMA/CA, the DCF is vulnerable to hidden node problems.

[5] Described in section 2.10.3
[6] Described in section 2.10.4

The PCF is not as widely used, and is commonly not implemented in current low-cost 802.11 equipment. The PCF uses a token-passing scheme, which prevents hidden node problems. However, the PCF is not applicable to 802.11 ad-hoc mode as it mandates the presence of an access point.

In conclusion, ad-hoc 802.11 networks use CSMA/CA and are thus susceptible to hidden node problems.

2.10.4 RTS/CTS : 802.11 Function to Alleviate Hidden Node Problems

Request-to-send and clear-to-send (RTS/CTS) optionally allows the access point to control the use of the medium in order to alleviate the hidden node troubles. A node wishing to transmit an amount of data may first transmit an RTS frame. The AP responds by broadcasting a CTS frame, instructing other nodes to wait for a period while the initial node transmits.

Because large frames take longer to transmit, they are statistically more likely to result in collision than smaller frames. Nodes are thus commonly configured to activate RTS/CTS for only large frames.

While RTS/CTS can help to avoid some hidden node problems, it does not eliminate hidden node problems. Because RTS frames are small by comparison to some payload data frames, they are less likely to collide than large data frames. However, in dense or busy networks the RTS frames can themselves cause collisions. This leads to severe performance degradation.

A study [29] reveals that the RTS/CTS scheme is not beneficial in most network scenarios (for example, for the 54 Mbit/s data rate as specified in the 802.11a standard) and that RTS/CTS effectiveness in improving throughput and average packet delay performance is uncertain

2.10.5 Multi-hop Ad-Hoc Networks with 802.11

In an 802.11 BSS, nodes are constrained to a small geographic area and can all communicate directly with one-another in ad-hoc mode. Because nodes can communicate with one-another directly, one hop is sufficient to reach any other node and no routing is required.

If the nodes in the BSS are moved outside of that small area, or if an obstacle is introduced, not all nodes will be able to communicate with one-another. In such cases, intermediary nodes can route data from the sender to the recipient via multiple hops.

Such networks are clearly possible, even though 802.11 was not designed to support them.

2.10.5.1 Multi-hop Networks

When a node in a network cannot communicate directly with all others, but a path can be found through adjacent intermediate nodes to any other node, intermediate nodes can forward data on behalf of the sender. This is facilitated by deploying an appropriate routing protocol.

When each node on the 802.11 network is also a router, the network is an 802.11 ad-hoc network.

2.10.5.2 Hidden Nodes in Multi-hop Networks

If all nodes in a network could communicate with one-another, there would be no need to multi-hop. In a multi-hop network, intermediate nodes forward data for nodes which cannot hear one-another. Because the source and destination nodes cannot hear each other, there is a high risk of hidden node collision at each of the intermediate nodes.

In other words, multi-hop networks tend to have a large portion of nodes which are hidden to other nodes.

Multi-hop ad-hoc networks built with 802.11 use CSMA/CA and are thus highly susceptible to hidden node problems.

2.10.6 Summary

The 802.11 DCF uses the CSMA/CA access method. CSMA/CA is vulnerable to hidden node problems, but the 802.11 MAC includes functions to alleviate these problems. These functions are only applicable to infrastructure networks with access points.

The 802.11 ad-hoc mode does not use access points and is thus vulnerable to hidden node problems.

Because multi-hop networks tend to have a large portion of nodes which are hidden to other nodes, such networks built with 802.11 are very susceptible to hidden node problems.

Chapter 3

Related Work

This chapter reviews previous work relating to the problem area..

3.1 Introduction

This chapter contains a survey of literature relevant and related to the problem areas which are the focus of this research.

3.2 TDM Above The MAC

3.2.1 Frottle : The Freenet Throttle

Frottle [15] is a GNU Open-source Linux application which was developed by members of the Melbourne Wireless Community to overcome hidden-node problems which were crippling their community network.

Frottle eliminates the hidden node effect on large scale wireless networks, by scheduling the traffic of each node so that one node does not swamp the Access Point. The scheduling of packets also prevents clients with stronger signals from receiving an unfair share of bandwidth, thus depriving those with weaker signals.

Frottle operates at the IP layer, simulating a token ring network. The Frottle tokens are passed using UDP datagrams. Access to the network is controlled by a master node. Clients send data

one-at-a-time, on receiving the token. This eliminates collisions. Since each client gets a limited slice of the bandwidth, clients get fair access regardless of their signal strength. Frottle effects a kind of time division multiplexing (TDM).

TDM is performed at Layer-3, using IPTABLES [6] under Linux. IPTABLES is typically installed on Linux machines to provide firewall functionality. It allows user-space applications to examine outgoing IP packets.

IPTABLES filters packets by applying a set of rules which direct traffic to various "targets". A QUEUE target is implemented, which is able to queue packets until such time as a user-space application examines and dispatches or discards them.

Whilst the mechanism results in increased latency, overall network performance and utilisation can significantly increase.

Frottle's token-passing scheme is similar to the 802.11 PCF. The PCF is not widely implemented in low-cost hardware networks, hence the need for Frottle.

Frottle is widely used. It resolves hidden-node problems and fairly shares bandwidth between nodes. Co-ordination is performed by a master-node, however, and it is only suitable for infrastructure networks.

3.2.2 WiCCP : Wireless Central Coordination Protocol

The Wireless Central Coordination Protocol (WiCCP) [22] is very similar to Frottle, but is reported not to provide as good performance.

Unlike Frottle, which is implemented as a user-space application, WiCCP is implemented as a kernel module. For this reason, WiCCP is technically more complex to deploy than Frottle.

WiCCP introduces an intermediate layer between the IP and MAC layers, providing Link-layer functionality to a secondary protocol stack. This approach means that WiCCP does not require an IP stack in order to function. Because no IP stack is required, this approach may suit embedded applications or sensor networks in which individual nodes have limited processing power and memory. Conversely, most networking applications today do require an IP stack. Thus there are few scenarios under which WiCCP's non-requirement for an IP stack is likely to be of any real advantage.

WiCCP is not stable, and development of the application appears to have come to a halt.

3.2.3 WiCTP : Token-based Access Control Mechanism for Wireless Networks

The Wireless Cyclic Token Protocol (WiCTP) [37, 36] is a research project developed at the University of Western Australia. The authors claim that WiCTP offers better performance than either Frottle or WiCCP.

WiCCT operates in an almost identical manner to WiCCP. It is not widely used or deployed.

3.2.4 Ad-hoc Frottle

The Melbourne Wireless Community, authors of Frottle, have initiated a project to develop a version of Frottle suitable for ad-hoc networks.

Ad-hoc Frottle [1] is still in the design phase at the time of writing. The proposed design utilises a token-passing scheme similar to that implemented in Frottle, but without the need for a central, coordinating master node.

Multiple tokens may be present in a network under the proposed design. This approach poses complex problems which the designers have yet to solve. Our opinion is that this approach is unlikely to yield a workable solution

3.2.5 Overlay MAC Layer (OML)

The Overlay MAC Layer [50] uses loosely synchronised clocks to divide time into slots of equal size, and uses a distributed algorithm to allocate these slots to nodes.

OML is unique in that it utilises a distributed slot allocation algorithm, and is thus suitable for ad-hoc networks.

Like WiCCP, OML inserts an intermediate network layer above the Data Link Layer, and beneath the Network Layer. The authors refer to this as an Overlay MAC layer.

OML does not directly address the hidden node problem, but it does provide a model for fair bandwidth allocation.

A crude synchronisation scheme is described in [50] which does not lead to accurate clock synchronisation, although the authors do point out that an improved synchronisation algorithm could be implemented. Because nodes are not accurately synchronised, time slots have to be relatively

large to account for error. Furthermore, the synchronisation scheme requires a master node and thus does not suit the ad hoc paradigm.

The OML scheme transmits a fixed number of packets during each slot. It does not address the issue of optimising slot utilisation.

Although OML consumes bandwidth because of its signalling overhead, it removes the need for nodes to contend with CSMA/CA. The loss of throughput to signalling overhead is approximately balanced by the gains of eliminating contention.

3.2.6 Summary

A strong case can be made for working at higher layers to solve the hidden node problems associated with the MAC layer. MAC functionality is generally encapsulated in the drivers and firmware of wireless devices. Such code is manufacturer-specific, non-portable and commonly not Open Source. This makes it very difficult to modify. Furthermore, any modifications would only apply to the specific hardware for which the driver or firmware was being updated.

By working at a higher layer, we can use a standardised interface such as IP provides. The implication is that the code could be written once and reused on multiple heterogeneous hardware platforms. Furthermore, the approach does not require any modification to drivers or firmware and can thus be deployed on existing hardware.

Frottle and WiCCP have been deployed successfully in community networks utilising Access Points. Both Frottle and WiCCP are only applicable in Access-Point scenarios, and are thus unsuitable for Ad-Hoc networks.

3.3 MAC Schemes

The Data Link Layer of the OSI stack is comprised of two sub-layers: the Logical Link Control (LLC) and the Medium Access Control (MAC).

The LLC provides a standard means of communicating with the Network Layer, while the MAC is responsible for formatting and transmitting Layer 2 frames.

The standardised IEEE 802.11 distributed coordination function (DCF) uses CSMA/CA, under which nodes are only allowed to transmit once they cease to receive any transmissions from other nodes.

3.3.1 ADHOC-MAC

ADHOC-MAC [28] implements a dynamic TDM mechanism that is able to provide variable-bandwidth reliable channels, needed for QoS delivery. Dynamic TDM is achieved by the Reliable R-ALOHA protocol (RR-ALOHA) [27], a distributed reservation protocol capable of dynamically establishing a reliable single-hop broadcast channel, which provides knowledge of MAC transmissions in overlapping segments. Unlike R-ALOHA, RR-ALOHA requires no central repeater and operates in a completely distributed manner.

3.3.2 SYN_MAC

SYN_MAC [55] is another slotted MAC scheme. It claims to outperform ADHOC-MAC, but requires network-wide synchronisation. Nodes reach system-wide synchronisation by detecting the pilot signals in cellular systems and/or the GPS signals from satellites.

3.3.3 FPRP - A Five-Phase Reservation Protocol

FPRP [58] is similar to ADHOC-MAC. The protocol performs the tasks of channel access and node broadcast scheduling, allowing nodes to make reservations within TDM broadcast schedules. It employs a contention-based mechanism by which nodes compete with each other to acquire the TDM slots.

3.4 Timing and Synchronisation

3.4.1 Introduction

This section discusses work relating to timing and synchronisation. Synchronous TDM requires that the clocks of nodes be accurately synchronised.

3.4.2 NTP and SNTP

The Network Time Protocol (NTP) [11, 41] is able to accurately synchronise the clocks of nodes to a common central clock. Simple Network Time Protocol (SNTP) [42, 43] is a simplified version of NTP.

NTP estimates delay time to be half of "the total delay minus remote processing time" [11], by timing the return-trip of a packet and assuming symmetrical delays. Once the delay time is known, the protocol is able to accurately synchronise one clock to another. However, NTP requires some reference clock that defines the true time in order to operate. The need for a central reference clock does not suit the ad-hoc paradigm.

3.4.3 Reference Broadcast Synchronisation

A synchronisation method for wireless broadcast networks, and especially ad-hoc networks, called Reference Broadcast Synchronisation (RBS) is discussed in [33].

The method relies on reference broadcasts or beacons, which do not contain explicit timestamps. Instead, the receivers use the arrival timestamp of the beacon as a point of reference for comparing and adjusting their clocks. RBS employs nontrivial mathematical methods.

3.4.4 Sensor Network Synchronisation

In [57], a MAC scheme for ad-hoc sensor networks is discussed. The scheme requires periodic synchronisation among neighbouring nodes to overcome the effects of clock drift rates, leading to a loose synchronisation between neighbouring nodes. The scheme is simple, and does not account for latencies in beacon transmissions.

3.4.5 Summary

The research described in this section shows that it is possible to achieve accurate node clock synchronisation in an ad-hoc network. We can draw on the work of other researchers should node synchronisation be required by this research.

3.5 Chapter Summary

This chapter reviewed various previous work relating to the problem area. We will draw on much of this work in order to provide a solution to the research objectives.

Chapter 4

Requirements Analysis

This chapter discusses and evaluates the requirements for a software application to facilitate the research objectives.

4.1 Research Objectives Reviewed

Our research objective is the design of a software mechanism, which:

1. Provides each node with a fair share of available bandwidth
2. Resolves Hidden Node problems
3. Is applicable to existing 802.11-based devices

These objectives are discussed in further detail below.

4.1.1 Fair Bandwidth Sharing Between Nodes

Section 2.7 discussed research which shows that the 802.11 MAC does not guarantee each node a fair share of bandwidth in a multi-hop network.

802.11 devices are able to transmit at various speeds, to allow for varying environmental conditions. There is thus no guarantee that the transmit rates of all nodes will be identical.

It is necessary to consider what we mean by "fair" bandwidth sharing. Given that different nodes can transmit different amounts in a fixed amount of time, should we seek to balance their throughput or should we allow each a fair share of time?

4.1.1.1 Equal Throughput

Heusse *et al* [39] describe a performance anomaly of 802.11 which occurs in seeking to balance throughput. The performance of 802.11 degrades as nodes match their flows to that of the slowest channel .

Furthermore, existing nodes might be required to reduce their throughput in order to allow for the addition of a slower node. Thus the bandwidth available to each node at any time is dependent on the activity of other nodes.

4.1.1.2 Equal Time

Conversely, if the objective is to provide each node with a fair share of time, all nodes are ensured an equal time share of channel access.

Nodes with higher transmission rates can obtain throughput increases independent of the channel qualities of others. Likewise, slower nodes are also guaranteed a fair time-share.

By offering each node an equal share of time, we achieve the desirable characteristic of performance isolation.

4.1.1.3 Time Division Multiplexing

TDM divides the capacity of a medium into time slots. Each slot is able to carry a separate data stream. Because the sender and receiver are synchronised, they are able to identify the slots and recreate the data streams.

The sender uses a multiplexer to place data into the slots. The receiver uses a demultiplexer to extract data from the slots.

TDM requires that the sender and receiver be accurately synchronised. A queue is also required, from which data can be dequeued during each slot's duration.

4.1.1.4 Summary

In order to provide performance isolation and overcome the performance anomaly, one should allocate each node an equal time slice rather than seek to balance their throughput.

TDM divides time into equal time slices, each slice carrying a different data stream. TDM thus meets our research objective of fairness.

4.1.2 Resolve Hidden Node Problems

Hidden node problems can be effectively overcome by a suitable slot reservation scheme. The reservation protocol is able to identify hidden nodes. It assigns slots so as to ensure that no two nodes on the network are allowed to transmit simultaneously such as to cause a collision.

4.1.2.1 Reservation Protocol

The reservation protocol enables a node to reserve repeating time-slices. The reservation scheme must prevent hidden node problems by allocating slots in such a way as to guarantee collision-free communication.

RR-ALOHA, FPRP and SYN-MAC require nodes to broadcast periodic beacon packets. These beacons contain information about the nodes within range of the broadcasting node.

4.1.2.2 Identification of Hidden Nodes

Hidden node problems involve at least three nodes. There have to be at least two nodes which cannot communicate directly with one-another, and a third node which can communicate with both the others.

Thus nodes which are reachable by two hops can cause hidden node problems to one-another.

For the purposes of the discussion which follows, we introduce two definitions:

- The **one-hop** (1H) cluster of a node consists of those nodes which can be communicated with directly by a given node. Exactly one “hop” is required to reach these nodes.
- A **two-hop** (2H) cluster of nodes consists of those nodes which can be reached by exactly two hops - in other words, one intermediate node is required to route information between a node and its two-hop cluster.

A node receives information of its 1H cluster directly, via the beacons which those nodes broadcast. Within each beacon broadcast, each node also transmits the IDs of its 1H cluster. In this way nodes are able to learn of their 2H clusters from the beacon broadcasts of their 1H clusters.

The set difference between a node's 2H cluster and 1H cluster is the set of nodes which are hidden to that node. Put differently, any nodes which are members of a node's 2H cluster and not of its 1H cluster are hidden nodes.

4.1.2.3 Summary

The reservation protocol is able to identify hidden nodes through the beacons it receives. It reserves slots so as to ensure that hidden nodes do not cause collisions.

4.1.3 Applicability to Existing 802.11 Devices

Under the OSI model, the obvious layer for TDM to take place is the MAC sub-layer of the Data Link Layer (DLL). However, the objectives of this research require that our solution be applicable to, and compatible with, existing equipment. This places constraints on the design of our solution.

4.1.3.1 Modifying Layer 2: The MAC Sub-Layer

The MAC is implemented in hardware, firmware and software [50]. Because the MAC interacts with hardware, it is device dependent. This makes it difficult, and potentially expensive, to modify the MAC.

Furthermore, a significant modification to the MAC would most likely render it incompatible with existing 802.11 devices.

Finally, there are existing MACs which solve the same problems which this research seeks to address. Our objective is not to replicate the work of others, but rather to discover whether such work can be applied to existing 802.11 devices.

Given that is is not feasible for us to intervene at the MAC sub-layer, we need to seek other points in the network stack where intervention might be possible.

4.1.3.2 Layer 2.5 : An In-Between Layer

[22, 37] and [50] propose an intermediate (overlay) layer.

The intermediate layer is between the Data Link and Network layer. This can be achieved by installing a virtual network adapter which utilises the layer-2 services of an existing NIC, and provides layer-2 services for a new network stack atop the virtual adapter.

This approach requires the installation of kernel modules under Linux, and may necessitate kernel recompilation.

4.1.3.3 Layer-3 Intervention: IPTABLES

An alternative approach is demonstrated by the Frottle (section 3.2.1) application. Frottle utilises IPTABLES to queue outgoing data, which it then dispatches at the appropriate time. Signalling data is transmitted using ordinary UDP packets.

Frottle is implemented as a user-space Linux application. This makes it simple to deploy.

4.1.3.4 Summary

A layer-2 intervention is inappropriate. Two alternatives are layer 2.5 and layer 3.

While Frottle works via IPTABLES, WiCCP does almost the identical job via an intermediate layer.

For reasons of easy implementation and deployment, this project will use a Layer-3 intervention via IPTABLES.

4.1.4 Summary

The fairness objective can be met by implementing synchronous TDM. A slot reservation algorithm is able to overcome hidden node problems. In order for the software to be applicable to existing devices, MAC layer modifications are not feasible. Thus we will intervene at Layer 3, using IPTABLES, to provide TDM functionality.

4.2 Other Considerations

The decision to implement TDM at Layer-3 raises further considerations. We discuss synchronisation and the payload.

4.2.1 Synchronisation

Time Division Multiplexing clearly requires accurate synchronisation between the clocks of nodes. A node joining an existing network will need to synchronise its internal clock to a common clock before it can reserve a slot. Furthermore, periodic re-synchronisation is required in order to compensate for drift, or small differences in accuracy between the clocks of individual nodes.

Section 3.4 introduced some relevant related work in the area of synchronisation.

This section investigates various clocks which could be employed to meet the synchronisation needs of this research.

4.2.1.1 External Clock

An external clock, such as provided by GPS receivers or GSM telephony networks, may be used. Nodes generally require additional hardware in order to use external clocks. The need for an external clock is undesirable because this research seeks to deliver a low-cost solution which is applicable to existing devices.

4.2.1.2 Time Service

Node clocks can be synchronised to an agreed central clock via a time service, such as is provided by protocols like NTP, the Network Time Protocol. However, a centralised service such as this does not suit the ad-hoc paradigm very well because ad-hoc networks can have dynamic topologies, and connectivity to any central node cannot be guaranteed when networks fragment.

4.2.1.3 Other-layer Synchronisation

In a given protocol stack, other layers might achieve synchronisation in which case the clock of that layer could be employed.

For example, in Infrastructure mode an 802.11 access point broadcasts periodic beacon frames. These are used by receiving stations to check their clocks. Because ad-hoc networks do not use access points, the 802.11 clock cannot be used for the requirements of this research.

4.2.1.4 Software Synchronisation to a Commonly Agreed Clock

It is possible to use software mechanisms to synchronise node clocks to a commonly-agreed, non-centralised clock. This approach does not require any additional hardware, does not depend on any centralised clock, and does not depend on the existence of clocks in other layers.

Such a scheme is described in [33]. A simplified scheme to achieve similar software synchronisation is described in section 5.5.1.

4.2.1.5 Summary

Because this research aims to provide a low-cost solution which is applicable to existing devices, synchronisation will be performed by software synchronisation to a common clock.

4.2.2 Transmission Mechanism

A difficulty of working at Layer 3 is that while the application can be aware of how much data it is transmitting, it does not know how long that data will take to transmit. This makes it difficult to calculate how much data a node can transmit within a single slot, without exceeding the slot.

This section investigates the timing difficulties.

4.2.2.1 Factors Affecting Transmission Time

The amount of time required to transmit a given amount of layer-3 data across a wireless network depends on factors such as:

1. Layer-2 Bit-rate
   This is the raw data rate at which the NIC can transmit. 802.11 devices are capable of operating at various rates. Faster rates allow more data to be transmitted in a unit of time.

2. Protocol Overheads
   The actual bit rate useable by higher layers will be less than the Layer-2 bit-rate as a result of the overheads added by the 802.11 MAC.

3. Contention Delays
   The CSMA/CA algorithm requires a node to wait until the medium is silent before transmitting. If the medium is not silent, the pending transmission would be delayed, thus reducing the amount of data that can be sent in a given amount of time.

4. Diminishing Random Back-Off
   802.11's DCF requires nodes which have finished a transmission to wait for a back-off period before attempting to transmit further (see section 2.6.3). This is to allow other nodes a chance to transmit.
   If the node senses that the medium is busy, it backs-off for a diminished period of time before trying again. The back-off period includes a random element.

It is possible to apply mathematical formulae to calculate the effects of factors (1) and (2) above, because they are deterministic. Our TDM scheme is able to guarantee that the medium will be silent, thus the impact of factor (3) can be predicted.

Factor (4), the Diminishing Random Back-Off, contains a random element. The impact of the diminishing random back-off thus cannot be accurately predicted.

The Diminishing Random Back-Off is only imposed if a previous packet has been transmitted within the time frame specified by the back-off period.

This means that the first packet which a node transmits during its slot will not be subject to (4), but subsequent packets transmitted during the slot will be.

The effects of (4) are small relative to (1) and (2), but compounded if multiple packets are transmitted one after another.

4.2.2.2 Slot Overruns and Underruns

The factors which affect throughput mean that it is impossible to know in advance how long a given amount of data will take to transmit. A node thus risks sending too much or too little data.

If a node transmits too much data during its slot, it will still be transmitting when its slot ends. This is known as a slot overrun. A slot overrun is a very bad condition because it can lead to

collisions with hidden nodes. Transmissions of other nodes may also be delayed, leading to further slot overruns, hidden node collisions, or a loss of synchronisation.

A node may, of course, transmit an amount of data which requires less time to transmit than the slot duration. This is called a slot underrun. Slot underruns are not error conditions. Slot underruns amount to unutilised capacity on the network.

Clearly, it is preferable for a node to err on the conservative side as underruns are preferable to overruns. At the same time, too large an underrun is inefficient.

4.2.2.3 Estimating Throughput

Although we cannot know in advance exactly how long a given amount of data will take to transmit, we can make an estimate.

We assume that:

1. The L2 transmit bit-rate of the node remains constant.
2. The protocol overhead is related to the size of the packet.
3. The medium is reserved for the transmitting node, thus the node does not need to contend and contention delays are eliminated.
4. The node has not recently transmitted another packet such as to invoke the diminishing random back-off.

If these assumptions hold true, then the time required to send a packet of fixed size will remain constant. This means that if we know how long packets of various sizes previously took to transmit, we can predict in advance how long future packets will take to transmit.

If assumption (4) does not hold true, we can predict to a lesser degree of accuracy how long future packets will take to transmit.

4.2.2.4 Methods of Estimating Payload Size

In order to make optimal use of a time slot, a node needs to estimate how much payload data can be transmitted without causing an overrun. If the assumptions in section 4.2.2.3 hold true, then the duration required to transmit a given packet can be estimated based on the packet size.

The following methods of estimating the amount of payload data which can be transmitted during a single slot are possible:

Predetermined - in which the network designer calculates and configures an appropriate (conservative) payload size in advance.

Formulaic method - the application may be given various parameters, such as the L2 line speed, and apply a calculation on these to estimate an optimal payload size.

Measured - the application can transmit a couple of test packets, and instruct other nodes to respond when they receive the packets. The application can use timing information in the responses to calculate the actual time that a packet took to transmit previously, and use this as an estimate of future payload capacity.

4.2.3 Methods of Implementing the Transmission Mechanism

This section describes three strategies which could be used to implement the transmission mechanism.

4.2.3.1 FIFO Queueing

A FIFO (First-in-First-Out) queue dispatches packets in the order in which they arrive. The Frottle application, described in section 3.2.1, uses FIFO queueing.

The token-passing mechanism used by Frottle removes the need for nodes to synchronise with one-another. However, for our purposes the FIFO approach approach results in two problems:

1. The approach is inefficient because if insufficient slot time remains to send the next packet, the packet cannot be sent. The remaining slot time would thus be wasted.
2. Each successive packet transmitted in this way compounds the Random Back-Off error described in 2.6.3.2. Because the application cannot accurately predict how long a series of packets will take to transmit, over-runs or large under-runs are likely.

This approach is workable if the application is very conservative with regard to how much data is transmitted during each slot, but is inefficient.

4.2.3.2 Packet Re-ordering

IP networks do not guarantee the order in which IP packets will be delivered [49]. The application thus theoretically has the freedom to re-order packets so as to achieve a more optimal slot utilisation.

This approach could mitigate the first FIFO inefficiency described above. However, the second inefficiency is still applicable and the random back-off periods tend to result in over-runs or large under-runs.

4.2.3.3 Packet Aggregation

The node could aggregate queued outgoing packets when available, so as to achieve an optimal payload. This is achieved by combining a number of small packets into one large packet.

Packet aggregation would improve the efficiency of the network because:

- The number of packets transmitted is reduced. This reduces the lower-layer protocol overheads.
- The application could size the aggregated data packet so as to make optimal use of the slot. Without aggregation, the last portion of most slots would be unutilised.

Because packet aggregation can combine multiple small packets into one large packet, it offers the potential to remove the effects of 802.11's random back-off period.

This would improve the efficiency of the mechanism, because the random back-off causes nodes to waste bandwidth while waiting.

The mechanism could further be improved because in the absence of the random back-off, more accurate calculations are possible thus over-runs and under-runs can be reduced.

Packet aggregation would require that all outgoing packets, regardless of their destination, be combined into a single broadcast packet. All receiving nodes would need to decode the broadcast in order to discover whether it contains packets destined for them.

A disadvantage of packet aggregation is that it requires additional processing from all nodes.

4.3 Summary

The requirements of fair bandwidth sharing and of overcoming the hidden node problem can be met by TDM and slot reservation schemes respectively. The slot reservation schemes require a slotted channel, such as provided by TDM.

So as to meet the requirement of being compatible with existing equipment, a Frottle-inspired Layer 3 intervention via IPTABLES is proposed.

TDM will require accurate timing, both for synchronisation and in order to optimise slot utilisation. Our design will utilise a software clock.

A FIFO queue can be implemented very easily, but is inefficient. The preferred means of implementing the transmission mechanism is packet aggregation.

The requirements are addressed in the next chapter, which introduces software designs to provide the functionality described in this chapter.

Chapter 5

Designs

This chapter introduces software designs to meet the requirements discussed in the previous chapter.

5.1 Introduction

The previous chapter concluded that the hidden node problem could be overcome by an appropriate slot reservation scheme such as RR-ALOHA. A slot reservation scheme requires a slotted medium, such as is provided by TDM. TDM also achieves the objective of fair bandwidth sharing.

This chapter proposes a layered design to facilitate slot reservation and TDM. The design abstracts functionality into distinct, modular components.

The chapter begins with an introduction to the various components of the design. This is followed by a description of beacon packets, which play a central role in the design.

The remainder of the chapter presents relevant design details for each of the components.

5.2 Functional Overview

The basic components of the design are briefly introduced below.

Time Divider The TD divides time into slots by implementing a timer. The timer fires an event at the start of each slot, to which the reservation protocol may respond. The time divider also provides facilities to synchronise its internal clock to an external clock.

Time Keeper The TK is primarily responsible for ensuring that the TD clocks of nodes remain synchronised. Using measurements of packet arrival times, the TK is able to calculate the beacon latency[1] of surrounding nodes. By compensating for this latency, the TK can estimate the time at which a packet was transmitted. Using this timing information, the TK can implement a distributed algorithm to synchronise the clocks of nodes.
The TK is able to estimate the duration of time required to transmit a given amount of data.

Multiplexer/Demultiplexer The MUX schedules and encodes outgoing transmissions, and decodes incoming transmissions. The MUX uses the TD to schedule outgoing transmissions to occur within the duration of a designated slot. It is able to accurately measure the arrival time of packets from other nodes, and provides this information to the TK.
The MUX encodes in-band signalling data in beacon packets, which are described later.

Reservation Protocol The RP is able to prevent hidden node problems by requiring nodes to reserve slots before they are allowed to transmit any data. The RP implements a distributed algorithm to manage slot reservations. It is able to identify hidden nodes, and to allocate slots in such a way as to prevent hidden node collisions.
The RP invokes the MUX to transmit data during specified time slots.

Queue (Q) is able to buffer outgoing IP packets. It enables the application to view the headers of these packets, and to dequeue packets for transmission.

The MUX uses the TD to facilitate Time Division Multiplexing. The TK ensures that the TD remains synchronised.

The RP manages reservations of the slots provided by TDM.

5.3 Beacon Packets

A beacon is a data packet which is broadcast at the start of each slot by the node which has reserved that slot. Beacon packets encapsulate signalling data, and provide timing information which is used to maintain synchronisation.

[1] Beacon latency is defined in section 5.5.2.3.

The beacon of each node has a constant data size and is assumed to take a constant amount of time to transmit.

A single beacon packet is transmitted during each slot. Bandwidth usage is optimised by combining all signalling information into this single packet.

The beacon is assembled and transmitted by the MUX. Table 5.1 provides an overview of the beacon information fields, some of which contain sub-fields which are managed by other components.

Field	Meaning
SI	Slot Index, managed by TK
SenderID	The unique address of the sending node
Timing	Timing information, used by the TK
Slot_table	Slot allocation table, managed by RP
Payload	Payload data

Table 5.1: Beacon Fields (Overview)

5.4 Time Divider (TD)

The time divider uses a timer to divide the capacity of the medium into time slots. It provides the following facilities:

5.4.1 OnSlot Event

The timer is programmed to fire an event each time it expires, signalling the start of each slot. The event handler notifies the TK of the start of a new slot via a function call, then invokes a handler in the RP. The RP handler checks whether the node is allowed to transmit, and if so invokes the MUX to transmit.

5.4.2 Timer Adjustment

In order to maintain synchronisation, nodes are required to adjust their internal clocks periodically. The time divider provides adjustment facilities which the nodes use to maintain synchro-

nisation to an external clock. The TK uses the timer adjustment facility to synchronise the TD clock.

5.4.3 Query the Timer

The TD provides functions which enable the application to query, in real-time, both the elapsed and remaining time for the current slot.

This information is used by the TK for the timing and latency calculations which are described later in this chapter.

5.5 Timekeeper (TK)

The TK provides various services for synchronisation and timing. It is able to synchronise the TD clocks of neighbouring nodes by accurately timing beacon transmissions. The TK provides the following services:

- Synchronise the TD clock by responding to beacon arrivals.
- Measure the latency of other nodes.
- Provide an estimation of the time required to send a packet of data of given size.
- Provide an estimation of the node's own beacon latency, and respond to notice of measurements from other nodes.

The latency measurement schemes described are based on those in [32].

5.5.1 Synchronisation

In order to be able to perform time division multiplexing, a node needs to be able to accurately synchronise its clock with the clocks of neighbouring nodes. There is no need for synchronisation with any real or central time.

Nodes achieve synchronisation by timing the arrival of beacon packets from other nodes. Beacon packets are broadcast by each active node at the beginning of that node's slot.

The proposed scheme leads to a local synchronisation, under which the clocks of nodes are synchronised closely to those within their two-hop cluster, but are not necessarily synchronised to nodes which are out of range.

Because some nodes are able to transmit with a lower latency than others, the beacon arrival times will be irregular. The receiving node is able to compensate for this if it knows the latency associated with the transmission.

5.5.1.1 Synchronisation Scheme

The synchronisation scheme attempts to synchronise TD clocks among nodes in a cluster. A node is required to synchronise with other nodes before it is allowed to transmit any data.

Over time, minor differences between the speeds of node clocks will lead to the phenomenon of clock drift. Nodes thus require periodic re-synchronisation.

The synchronisation scheme attempts to synchronise all nodes to the **fastest** clock on the network. This approach has the following benefits:

- A beacon transmission can be delayed, but it cannot be accelerated. A beacon cannot be received any faster than the beacon latency allows. A delayed beacon transmission will be understood by other nodes as a slow clock, to which they will not synchronise, and the delay will thus not disrupt synchronisation unnecessarily.
- It provides a simple, distributed method to agree a common clock when clusters merge, as Ad-hoc networks may become fragmented or associated.

Each time a beacon packet is received, the MUX notifies the TK of the beacon arrival. The arrival time of the beacon packet is stored as a timestamp.

Received timestamps need to be adjusted to compensate for the latency of the sender. Latency is discussed later in this chapter.

The synchronisation logic is elementary. When a beacon is received, if the timestamp indicates a time earlier than the receiver's clock, the receiver adjusts its clock to match. This results in nodes on the network agreeing on a common clock.

5.5.1.2 Network Fragmentation

A network becomes fragmented when a group of nodes become disconnected from the main network. The disconnected group becomes an independent network. This might happen if a node which routes data between two different clusters of nodes relocates or fails. This is a common occurrence in a MANET, because the topology is very dynamic.

Because the Layer-3 Reservation TDM scheme is distributed, it requires no central infrastructure. Thus network fragmentation poses no difficulties with regard to synchronisation.

5.5.1.3 Network Association

Ad-hoc networks can conversely associate, when connectivity is established between two previously independent networks via one or more common nodes.

There is no guarantee that the clocks of the two groups of nodes will be synchronised when the networks associate.

Fortunately, the node synchronisation routines (section 5.5.1) will result in the entire network adopting the clock of the faster network.

There will be a period of disruption, during which nodes may lose their slots, as they re-synchronise to the new clock.

A problem is posed if the configuration parameters of the two networks (the slot count and duration) differ. In order to avoid hidden node problems and interference, the networks **must** adopt a common configuration. It may be the case that the configuration parameters of one network are not suitable for the other.

Because a common configuration is required, and because the faster network clock is adopted, it may make sense for the configuration parameters of the faster network to be adopted too.

If two adjacent networks are planned with different configurations, they should use different 802.11 channels to avoid problems of this sort.

5.5.1.4 Slot Index (SI)

The Slot Index (SI) is a variable integer representing the number of slots which have elapsed since the first nodes in the network began transmitting. The SI can be multiplied by the slot duration to calculate the temporal age of a given network.

The SI is contained in every beacon packet. The transmitting node is responsible for calculating the SI when assembling the beacon packet.

A node's clock can be seen to have two components: The TD clock and the SI. The SI counts entire slots, while the TD measures fractions of a slot. The two can be thus be combined into a real number, which can be used for comparing node clocks.

5.5.1.5 SI and Association

The SI can be used to synchronise two entire networks when they associate. It provides a means of deciding which network's clock the nodes should adopt.

Any node which receives a beacon broadcast with an SI higher than its own must synchronise to the received beacon, and adopt the SI of the sender. Precedence is thus granted to the network clock with the higher SI.

5.5.1.6 Summary

The TK creates a timestamp each time a beacon packet is received. The timestamp is constructed from the SI and the TD clock.

The synchronisation scheme attempts to synchronise all nodes to the fastest clock on the network. If a received timestamp is earlier than the receiving node's clock, the receiver must re-synchronise to the received timestamp.

The next section discusses beacon latency, which must be taken into account when calculating the timestamp.

5.5.2 Measuring Beacon Latency

Because beacons are used for synchronisation, the synchronisation routines need to know the latency associated with the beacon transmission. Various methods are proposed which are able to provide latency estimates to varying degrees of accuracy.

5.5.2.1 Introduction

The beacon latency of a node in an 802.11 network is of the order of milliseconds, which is sufficient to skew any time-based calculations which nodes might perform. Thus it is important

for nodes to compensate for latency.

In order to synchronise, a node needs to know the latency of the node to which it is attempting to synchronise. The receiving node adds this latency to the arrival time of the beacon, and uses the sum as a timestamp for synchronisation.

5.5.2.2 Contiguous vs Non-contiguous Slots

For the sake of simplicity, the calculations which follow assume that beacons are received in contiguous slots. We assume that node **A** occupies the first slot, while **B** and **C** occupy the second and third slots, respectively.

Because packet arrivals are measured in terms of the TD clock, they are measured relative to the start of the current slot. Thus the calculations which follow are equally applicable if the slot reservations are not contiguous.

5.5.2.3 Beacon Latency

For the purposes of this document, we shall define the actual beacon latency, l_{ax} of a node **X** as the time difference between the moment when node **X** begins transmitting a beacon, and the moment when other nodes receive that beacon. Beacons have a constant byte size.

Because beacons are broadcast, it is fair to assume that receiving nodes will receive the beacon simultaneously.

5.5.2.4 Assumptions

- We assume that the beacon latency of a node remains constant.
- We do not make allowances for variable processing latencies on nodes. At the time when the latency measurement is made, any processing latency is included in the measurement. It is assumed that the variance of such latencies is negligible for our purposes.

5.5.2.5 Two Nodes: Sum of Latencies

If node **B** has previously synchronised to node **A**, the sum of their latencies can be determined by **A** at the moment when **B**'s beacon arrives:

Preconditions: **A** is a node, such that:

1. **A** broadcast a beacon at time t_0.
2. **B** synchronised to **A**'s beacon at time t_{ra}, which is not known to **A**.
3. **B** synchronised to **A** using latency l_{sb} (B calculates l_{sb} and announces l_{sb} in its beacon).
4. **A** received a beacon from **B** at time t_{rb}.

Postconditions: **A** learns the sum of the actual latencies of both nodes, $l_{aa} + l_{ab}$, and can thus calculate either in terms of the other.

Description:

1. At time t_0, the start of **A**'s slot, **A** broadcasts a beacon.
2. **B** receives the beacon after **A**'s send latency, thus $t_{ra} = l_{aa}$.
3. **B** synchronises to the beacon. If d is the slot duration, **B** waits a duration of $d - l_{sb}$ until the start of the next slot.
4. **B** transmits a beacon at the start of **B**'s slot.
5. **A** receives the beacon after **B**'s send latency at t_{rb}:

$$t_{rb} = l_{aa} + (d - l_{sb}) + l_{ab} \tag{5.1}$$

6. By rewriting 5.1, **A** solves for the sum of the latencies:

$$t_{rb} - (d - l_{sb}) = l_{aa} + l_{ab} \tag{5.2}$$

$$l_{aa} + l_{ab} = t_{rb} - (d - l_{sb}) \tag{5.3}$$

5.5.2.6 Two Nodes: Half Return Time (HRT) Method

Although two nodes can easily calculate the sum of their latencies, they cannot easily determine their respective individual latencies. In a homogeneous network, it is fair to assume that the send latencies of two nodes are equal. Thus, like NTP, we assume symmetrical latencies and halve the sum of the latencies.

Preconditions:

1. It is assumed that the beacon latencies of both nodes **A** and **B** are equal, thus $l_{aa} = l_{ab}$
2. The remaining preconditions are the same as for section 5.5.2.5.

Postconditions: **A** derives estimates for both l_{aa} and l_{ab}.

Description:

1. If we assume that the latency is symmetrical, then $l_{aa} = l_{ab}$. We can then rewrite 7.1 as:

$$l_{ab} + l_{ab} = t_{rb} - (d - l_{sb}) \tag{5.4}$$

$$2 * l_{ab} = t_{rb} - (d - l_{sb}) \tag{5.5}$$

$$l_{ab} = \frac{t_{rb} - (d - l_{sb})}{2} \tag{5.6}$$

to derive values for $l_{aa}$and l_{ab}.

5.5.2.7 Measuring Latency: 3 or more Interconnected Nodes

A node **C** can measure the actual beacon latency of a node **B** provided that node **C** received the beacon from node **A** to which **B** synchronised.

Preconditions: **C** is a node such that:

1. A beacon has been received from node **A** since **C**'s last beacon broadcast, at time t_{ra}.
2. **C** has just received a beacon from node **B**, at time t_{rb}.
3. Node **B** has synchronised to node **A** using an estimated latency l_{sb}.

Postconditions:

1. **C** calculates l_{ab}, the actual latency of node **B**.

Description:

1. At start of **A**'s slot, **A** broadcasts a beacon.
2. **C** receives the beacon at time t_{ra} after **A**'s actual beacon latency l_{aa} has elapsed. **B** receives the beacon simultaneously.
3. **B** synchronises to the beacon, waiting a duration of $d - l_{sb}$ until the start of the next slot
4. **B** transmits a beacon at the start of **B**'s slot.
5. **C** receives the beacon after **B**'s send latency at time t_{rb}:

$$t_{rb} = t_{ra} + (d - l_{sb}) + l_{ab} \tag{5.7}$$

6. **C** solves for l_{ab}:

$$l_{ab} = t_{rb} - t_{ra} - d + l_{sb} \tag{5.8}$$

5.5.3 Propagation of Latency Information

It is desirable for nodes to learn of their own latencies, so that they can broadcast this information in their beacons. Other nodes can use the latency information received in beacons to synchronise to the sender.

5.5.3.1 Propagation Method

A constant portion of the beacon is reserved for use by the timekeeper (TK). This allows the protocol to maintain a constant signalling overhead. A single field is provided, called the Latency Notification field, which allows the node transmitting the beacon to notify a single other node of that node's beacon latency.

Latency information is propagated in a "most-needed" manner. Each node transmits what it believes to be its own latency in a field in the beacon. Nodes which are able to calculate the latency of the transmitting node compare their calculation to the beacon field. The absolute value of the difference between the latency value of the beacon and the calculated latency is known as the Latency Error.

$$e(l) = |l_{sx} - l_{ax}| \tag{5.9}$$

When a node assembles its beacon packet, it uses the Latency Notification field to notify the node with the highest Latency Error of that node's calculated latency.

Nodes also broadcast their calculation of the latency of the nodes to which they are synchronised. There is thus a mechanism which can be used to inform the node to which others synchronise, of its latency. The node to which other nodes will synchronise is the one with the fastest clock.

When a node synchronises to another, it announces the latency estimate which it used in its beacon packet. For this reason, the fastest node in the cluster can learn of its latency without any need for the latency propagation mechanism.

If the fastest node has the largest Latency Error, then the node with the second largest LE may be notified instead via the Latency Notification field.

5.5.3.2 Comparing the Accuracy of Estimates with the i-value

Nodes are able to measure or estimate the beacon latency of other nodes, but not their own. Nodes are informed of their latencies through messages which are sent by other nodes.

This section has described various ways in which latency might be measured. It is possible that a node will receive multiple, conflicting calculations of its beacon latency from surrounding nodes. Some methods of calculating beacon latency are more accurate than others, thus it is important for the receiving node to know the method which the sender used in calculating the estimate. In this way, the receiver can select the most accurate result.

For this purpose, we introduce a value i_x which indicates the accuracy of the corresponding estimate e_x. Latency information thus comprises a tuple: latency and an indicator of the method used to derive it. The i-values are described in table 5.2.

When a node receives multiple latency tuples from multiple nodes, the node should favour the latency with the lowest i-value.

Given a choice of different estimates C={c1, c2, c3 .. cn} representing a latency l_x, a node should choose the lowest i- value from C as representing the best estimate of latency.

Should there be one or more members of C with the same i value, the most recently received estimate should be used.

i-value	Meaning
10	3-Node Measurement
20	Half Return Time (HRT) Estimate
100	Initial Guess
+1	Add 1 if Sum-of-Latencies is used
0	Pre-calibrated (Reserved)

Table 5.2: i-value Meanings

5.5.3.3 Encoding the i-value Tuple Efficiently

The accuracy indicator i_x may be efficiently encoded by multiplying it by d (the slot duration) and adding it to e_x, yielding a compound estimate c_x such that

$$c_x = d * i_x + e_x \tag{5.10}$$

The receiver divides c_x by d. The integer result of the division is i_x , and e_x is the remainder.

Equation 5.10 assumes that the latency estimate is less than the slot duration, thus $e_x < d$.

5.5.3.4 The i-value and Assumed Accuracy Losses

A node can use the method described in section 5.5.2.5 to calculate one latency estimate based on another.

Any node which performs such a calculation to estimate e_y based on e_x should calculate i_y such that:

$$i_y = i_x + 1 \tag{5.11}$$

In other words, if a node uses an existing estimate of latency in a latency calculation, the i-value should be incremented to indicate an assumed loss of accuracy.

5.6 Multiplexer/Demultiplexer (MUX)

The MUX schedules and encodes outgoing transmissions, and decodes incoming transmissions. These tasks are called multiplexing and demultiplexing, respectively.

The reservation protocol requires a node which has reserved a slot to transmit a beacon at the start of that slot. Before transmitting any payload data during a slot, the MUX must thus assemble and transmit the beacon.

Beacon transmission, multiplexing and demultiplexing are discussed below.

5.6.1 Assembly and Transmission of Beacon Packets

Although the multiplexer itself does not use the beacon in any way, it is responsible for assembling and transmitting the beacon packet.

Beacon packets contain signalling information, used by components such as the RP and TK. For efficiency reasons, the signalling data of these components is combined into a single packet. This is because multiple small packets would require more overhead to transmit than one large packet of the same aggregate payload size.

Beacon packets also carry implicit timing information, which is used for synchronisation.

When assembling the beacon packet, the MUX combines information obtained from other components into a memory buffer, ready for transmission.

Beacon packets are transmitted as UDP broadcasts.

5.6.2 Multiplexing and Demultiplexing

When multiplexing, the MUX places outgoing data into a slot designated by the RP.

The RP manages slot reservations, and invokes the transmit routines in the MUX at the start of a reserved slot.

As discussed in the previous section, the first task of the MUX is to assemble and transmit the beacon packet. The beacon is followed by payload data.

Section 4.2.2 discussed various ways to optimise payload transmissions. The prototype application will address two different approaches, namely FIFO queueing and packet aggregation. The third approach discussed in section 4.2.2, packet re-ordering, is an enhancement of FIFO queueing and will not be addressed.

By addressing both FIFO queueing and packet aggregation, we are able to evaluate the two approaches and select a preferred one.

Naturally, the demultiplexing method must match the multiplexing method. On arrival of a beacon packet, the demultiplexer generates an event to notify both the TK and the RP of the arrival.

5.6.2.1 FIFO Queueing

FIFO Queueing is the approach used by Frottle. Under this approach, outgoing network-layer packets are queued until the beginning of the node's slot.

At the start of the node's slot, the MUX assembles and transmits a beacon packet. This is followed by an appropriate number of data packets.

A problem with this approach is that each packet transmitted after the beacon will invoke 802.11's diminishing random back-off. The random delay means that a node cannot accurately predict in advance how long a packet will take to transmit. It is thus difficult to estimate how much data can be sent during a slot without causing an over-run.

The FIFO queue is configured to transmit a fixed number of packets during each slot. Each packet can have a maximum size specified by the MTU. A slot duration must be configured which is large enough to allow all the packets to be transmitted, with an allowance for the random back-off. Configuration of slot duration is discussed in chapter 8.

5.6.2.2 Packet Aggregation

Packet aggregation combines multiple small packets into a single large packet in order to reduce protocol overheads.

Multiple IP packets can be aggregated by writing them into a stream of data. The individual IP packets can be extracted from the stream by reading the IP header fields, which specify the size of each packet.

In order to ensure that only one packet is transmitted during each slot, the beacon packet is enlarged to allow it to contain aggregated packets. The transmitting node copies an appropriate number of bytes from the IP data stream and transmits these in the beacon.

An important advantage of packet aggregation is that it has the potential to eliminate 802.11's diminishing random back-off. Because the node only sends a single packet during each slot, and because the slot is reserved for that node's use, the diminishing random back-off should never be invoked.

The latency measurement routines (described in this chapter) require the size of the beacon packet to be constant. Thus if insufficient outgoing packets are queued to transmit, the MUX must pad the beacon packet with dummy data.

Packet aggregation is more difficult to implement than FIFO queueing. The FIFO queueing approach only requires the transmitting node to schedule its outgoing transmissions appropriately. Packet aggregation requires the transmitting node, in addition to scheduling transmissions appropriately, to modify outgoing and incoming data.

It is easy to assemble and disassemble aggregated packets, but IPTABLES provides no mechanism for injecting these packets back into an IP stack.

Packet aggregation essentially necessitates translation between two different data link layers. Such gateway functionality is commonly offered by routers. A router with a virtual network device, could facilitate packet aggregation.

In order to limit the scope of this research, we will not seek to implement a router. Instead, the MUX will simulate packet aggregation by padding the payload field with dummy data.

5.7 Reservation Protocol (RP)

The RP manages node reservations to control bandwidth allocation and prevent hidden node problems. Various reservation protocols were discussed for the MAC schemes, described in section 3.3.

The application will implement a simplified version of RR-ALOHA, as discussed in the previous chapter. The simplified RR-ALOHA is described in this section. The section first introduces the Slot Table, then describes the Slot Reservation mechanism. A state machine is then described, which models the protocol.

5.7.1 Slot Table

The reservation protocol requires each node to maintain a slot table, which is transmitted with the beacon. The slot table reports the state of each of the slots as perceived by the node. Each slot may have one of three states: FREE, BUSY or RESERVED.

The slot table implements a sliding frame. Thus slot 0 is always the current slot, while slot 1 is the most recently elapsed slot, etc.

State is updated after each slot expires, as follows :

- If no beacon packet is received, the slot is marked FREE.
- If a beacon packet is received and successfully decoded, the slot is marked BUSY and the identity of the transmitting node is stored.
- Any BUSY slots in the received table are marked RESERVED in subsequent slots in the receiver's table.

This mechanism allows a node to learn of all slot reservations within its two-hop (2H) cluster. The slot reservation mechanism, described next, uses this information to ensure that slots are allocated so as to avoid hidden-node collisions.

5.7.2 Slot Reservation

Nodes contend to obtain their slots. Any node **R** can attempt to reserve a FREE slot, by transmitting a beacon at the start of that slot.

All nodes which receive **R**'s beacon mark the slot as BUSY by **R**.

If any beacon is received by **R** during subsequent slots which does not mark the slot as BUSY by **R**, the slot reservation is deemed unsuccessful. After a failed reservation, **R** relinquishes the slot and must attempt another reservation.

At the return of **R**'s slot, after a full frame of slots has elapsed, the reservation attempt is deemed successful.

Having successfully reserved a slot, **R** subsequently uses that slot to transmit information.

5.7.3 Slot Loss

A node **S** which has successfully reserved a slot can lose the slot.

If any beacon is received from another node by **S** which does not mark the slot as BUSY by **S**, the slot reservation is deemed lost. After a lost reservation, **S** must attempt another reservation.

Slot loss can have various causes. If any node **N** in **S**'s one-hop (1H) cluster fails to receive **S**'s beacon broadcast, that node will mark **S**'s slot FREE in its table. When **S** receives **N**'s beacon,

S will be forced to consider its slot lost. Slot loss can thus be caused by data communication errors.

It can also be caused by synchronisation errors, if the TD clocks of two nodes disagree.

Changes in physical topology might lead to two nodes which previously reserved a common slot moving within each-other's 2H cluster. Their reservations would conflict, and one or both nodes would lose their slots.

Nodes can voluntarily relinquish their slots by ceasing beacon transmissions.

5.7.4 RP State Machine

The RP is modelled as an event-driven finite state machine. A state-change diagram is provided in fig 5.1. This should be read in conjunction with tables 5.3 and 5.4 which define the states and transitions between those states respectively.

5.7.4.1 State Definitions

Table 5.3 defines the possible states of the RP.

State	Name	Meaning
S1	ClockInit	The timer needs to be calibrated - see section 6.2.3.1.
S2	Unsynchronised	The application was recently started. No beacon has yet been received.
S3	Synchronised	A beacon has been received from another node. The node's TD clock is now synchronised to another.
S4	WantSlot	The application requires a slot for the basic channel.
S5	AttemptingReservation	An available slot has been found, and the the node is waiting to attempt a reservation.
S6	AwaitingReservation	The node is waiting to ensure that all surrounding nodes acknowledge the reservation.
S7	GotSlot	The performance measurements are completed, and the node is free to transmit payload data.

Table 5.3: State Definitions for Reservation Protocol

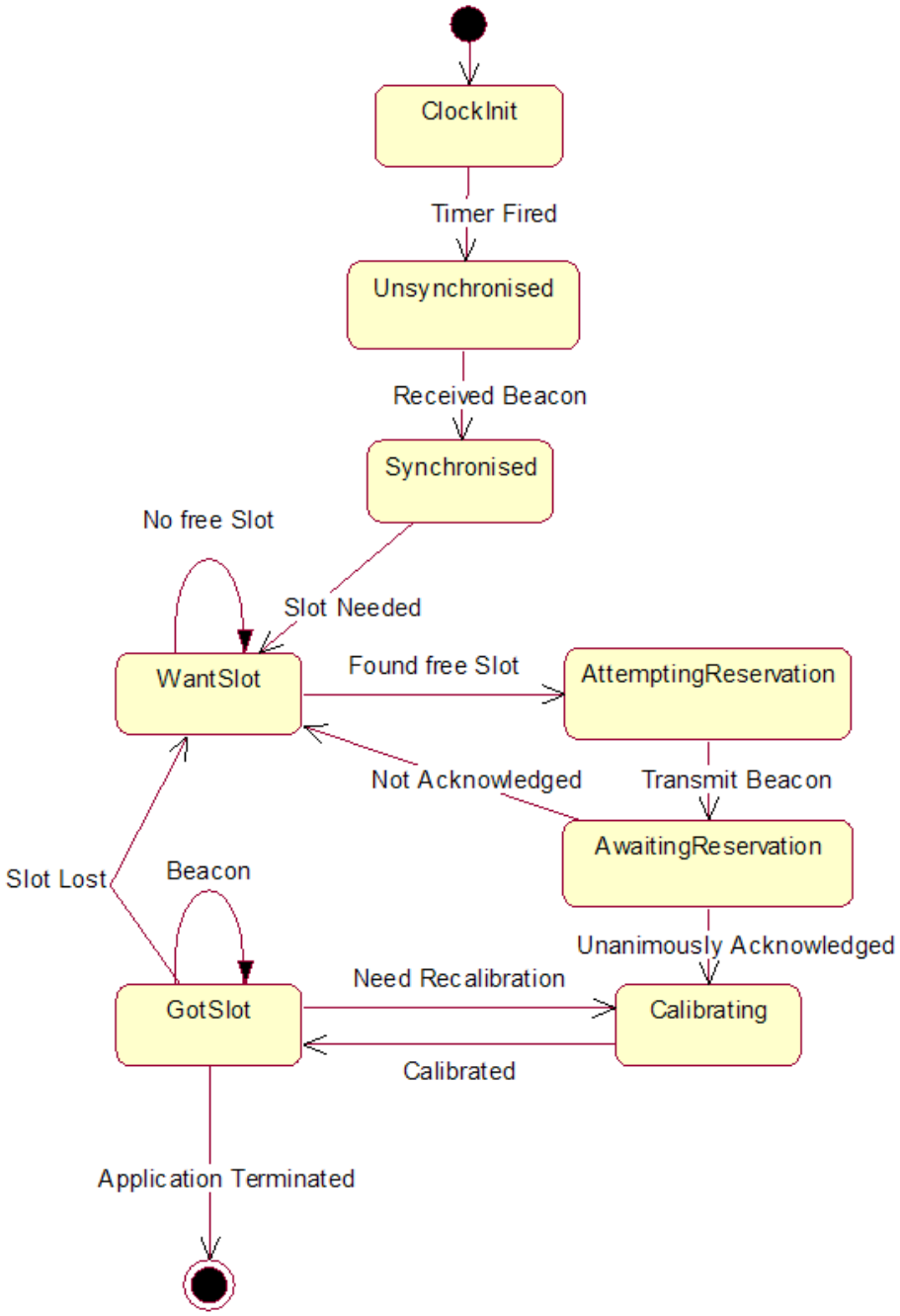


Figure 5.1: Reservation Protocol State Change Diagram

5.7.4.2 State Transitions

Trans	Name	Description
T1	Timer Fired	Initial timer event for calibration - see section 6.2.3.1.
T2	Received Beacon	A beacon has been received from another node, to which the application can synchronise its clock.
T3	Slot Needed	The application has requested the RP to attempt to reserve a slot.
T4	Found Free Slot	A free slot was found in the slot table.
T5	Transmit Reservation	The beacon has been transmitted, attempting a reservation.
T6	Not Acknowledged	A beacon message has been received which does not acknowledge the reservation attempt.
T7	Success	A beacon was received acknowledging this node's slot reservation.
T8	Slot Lost	A beacon message has been received which does not acknowledge this node's reservation, thus it must assume that slot has been lost.
T9	No Free Slot	The RP searched, but was unable to find a free slot and will try again later.
T10	Reservation Acknowledged	A full frame has passed since the reservation was attempted, and all received beacons have acknowledged the reservation.
T11	Application Terminated	Application terminated requested (simplification).

Table 5.4: State Transitions for Reservation Protocol

Figure 5.4 lists the transitions between states for the RP. The RP is entirely event-driven, and thus does not require its own thread of execution. The two events to which the RP responds are the OnSlot event, generated at the start of each slot by the TD, and the OnBeacon event, generated each time the MUX receives a beacon.

The state transitions in table 5.4 must thus be mapped to these two events.

5.8 Queue (Q)

The Queue is used to buffer outgoing IP packets. It enables the application to view the headers of these packets, and to dequeue packets for transmission. The queue provides the following functionality:

- Queue outgoing data.
- Enable application to view headers of queued packets.
- Prioritise data according to predefined rules.

5.9 Summary

This chapter described software designs. A virtual TDM mechanism was designed, which required accurate synchronisation. Synchronisation requires compensation for latency. A slot reservation protocol is implemented over the virtual TDM.

The designs are implemented and tested in the prototype application, which is described in the next chapter.

Chapter 6

Implementing the Prototype Application

This chapter describes how the various components of the prototype application were implemented. We draw attention to the various issues which were identified during implementation, and describe how the issues were overcome.

6.1 Development Environment

The prototype application was developed using the standard GNU C compiler. It compiles and runs under Linux, and has been successfully tested on 2.4.x and 2.6.x kernel versions. Ubuntu Linux, a derivative of the Debian Linux distribution, was used for development and testing.

6.2 Functional Components

This section describes the implementation of the various components of the application. We draw attention to the various issues which were encountered while implementing each component.

6.2.1 Time Keeper (TK)

The TK implements a function called `OnBeaconArrival()`. This function is called by the application each time a beacon packet is received.

A function called `CalcLatency()` is called by `OnBeaconArrival()` to calculate the latency of nearby nodes, using information received in the beacon. `CalcLatency()` implements the latency measurement algorithms described in section 5.5.1.

6.2.1.1 Issue: Process Latency and Tagging Packet Arrival Time

Our code times the arrival of packets. However, if the Linux scheduler defers the execution of our code, timing is only performed at the start of our time-slice and inaccurate measurements result.

This issue was noted when we attempted to reduce the slot size from 100ms to 10ms, which is of the order of a Linux time-slice. At this resolution, the error introduced by process latency caused the application to repeatedly lose synchronisation and thus slot reservations.

The Linux kernel provides a mechanism which can timestamp a packet as it is received from the physical interface. This is used by programs such as **`ping`** and **`NTP`**.

A function called `getProcessLatency()` was implemented, which calculates the time difference between a given timestamp and the current system time. The return value of this function can be used to compensate for process latency in calculations.

The use of this mechanism greatly improves accuracy, and thus improves the performance of the application.

6.2.2 Reservation Protocol (RP)

The reservation protocol implements a simplified version of the RR-ALOHA protocol. This is described in section 2.6.4.

6.2.2.1 Issue: Random Slot Reservations

When the reservation protocol was initially implemented, we programmed nodes to contend for the first available slot after achieving synchronisation with other nodes.

This approach leads to very slow network startups, because if nodes were started simultaneously they would all contend for the same slot. Only one node could win each round, and the remaining nodes would then contend again for another slot.

By programming nodes rather to contend for a random available slot, this problem was alleviated and network start-up time was improved.

6.2.3 Time Divider (TD)

The TD implements a POSIX Interval Timer (ITIMER) to divide time into slots. The timer expires at the start of each slot, and generates a signal (SIGUSR1). A handler function traps the signal and passes control back to a handler in the application.

Two functions, `getSlotRemaining()` and `getSlotConsumed()` are provided to query the ITIMER to read the remaining or elapsed slot durations respectively.

A TD function called `Synchronise()` is called by the TK in order to synchronise the TD clock to those of the rest of the network when the beacon latency of the sending node is known.

The `Synchronise2()` function accepts a timestamp. If the operating system is able to provide an arrival timestamp for a packet, this can be used as described in section 6.2.1.1.

6.2.3.1 Issue: POSIX Timer Calibration

POSIX ITIMERs are set using the `timer_settime()` function. An ITIMER is able to provide very accurate timing, exceeding the resolution of the underlying hardware timer.

In order to provide accurate timing, the ITIMER profiles CPU execution speed. It is able achieve a resolution higher than the hardware timer by using CPU cycles to compensate.

This mechanism is calibrated the first time an ITIMER is used. For this reason, the initial call to the `timer_settime()` function may yield inaccurate results. Subsequent calls to the `timer_settime()` function behave as expected.

This problem was remedied by programming the application to ignore the first ITIMER expiration.

6.2.3.2 Issue: Timer Over-Runs

Because modern operating systems time-slice in order to multi-task [44], it is possible that another process will be using the CPU when the timer fires.

This became a problem when we discovered that other processes were deferring the execution of our process, after the timer expired.

When a timer expiration signal is delivered to or accepted by a process, if the implementation supports the Realtime Signals Extension, the `timer_getoverrun()` function returns the timer expiration overrun count for the specified timer. The overrun count returned contains the number of extra timer expirations that occurred between the time the signal was generated (queued) and when it was delivered or accepted [45].

Thus, if the `timer_getoverrun()` function returns a non-zero value, the application becomes aware that an over-run has occurred. The application is then too late to transmit the beacon at the appropriate time.

Timer over-runs should not be confused with slot over-runs, described in section 5.5.1.

When a timer over-run occurs, the node is required to abandon its slot. The node must then contend again to reserve another slot.

The probability of a timer over-run was discovered to be greater for a node with a slow CPU than for a node with a fast CPU. The probability of timer over-runs increases when the load on the CPU increases, as happens when additional application processes are spawned.

The prototyping network consisted of Pentium-4 class machines running Ubuntu Linux. In single-user text-only mode, with no other applications running, slot over-runs were rare (less than 2% of slots were over-run). With a graphical user interface (GUI) and some desktop applications running, slot over-runs occurred alarmingly often (more than 20% of slots were over-run).

6.2.3.3 A Remedy: Process Priority

Timer over-runs can be reduced by boosting the process priority of the application. This is because the process will then receive priority over other processes.

The Linux `nice` program is used to set process priority. To start the application in super-user mode with maximum priority, a command such as this is used:

```
sudo nice -n -20 ./slottle
```

By boosting the priority of our process to the maximum, using `nice`, overruns are significantly reduced and performance is significantly improved.

By using `nice` on the prototyping network, we were able to reduce the probability of slot over-runs when a GUI and other applications are running. The use of `nice` reduced the frequency of over-runs to an acceptable level, similar to that experienced when running in single-user text-mode.

6.2.4 Queue (Q)

The source code for the Q is based on that used in the Frottle application.

The Q implements a POSIX thread to interrogate IPTABLES about new outgoing datagrams.

Beacon packets are not kept in the queue, but dispatched immediately because they contain time-sensitive information. Other packets are sorted into three prioritised queues.

No major issues were encountered while implementing the Q.

6.2.5 Multiplexer (MUX)

The MUX implemented in the prototype application is a simplified one, intended for experimental rather than real-world use.

The MUX transmits a beacon packet at the start of each slot, followed by a statically configured number of packets. No demultiplexer is required in this simplified MUX.

A function called `SendPackets()` is implemented to transmit packets. This is called by a timer at the start of a node's slot. `SendPackets()` removes packets from the queue and transmits them.

6.3 Summary

This chapter described the implementation of the components of the application. We drew attention to the various issues which were encountered while implementing each component, and explained how each was addressed.

Chapter 7

Experiments

In this chapter we document the experiments which were performed during the course of this research.

7.1 Introduction

This chapter describes three experiments which were performed in order to address questions which arose during this research.

A preliminary experiment was performed in order to discover whether the amount of time required to send a constant amount of IP data remains constant. This is important in order to be able to predict the amount of time required to transmit a given amount of data.

A further experiment was performed to evaluate the range of the diminishing random back-off period in the inter-packet gap.

A final experiment measures beacon latency for nodes in our network. Measurement of beacon latency is important in order to correctly configure a node, as well as in evaluating the performance of the application.

Type	Compaq nx9010 Notebook
IP Address	192.168.1.100
Processor	2.8 GHz Pentium 4 CPU
Memory	1024Mb RAM, 40Gb HDD
Network Adapter	Compaq WL110 PCMCIA 802.11b Wireless Adapter
Operating System	Ubuntu Linux, Kernel version 2.6.12-9-386

Table 7.1: Node #1 Configuration

Type	Configuration
IP Address	192.168.1.98
Processor	3.0 GHz Pentium 4 HT CPU
Memory	1024Mb RAM
Network Adapter	D-Link DWL-520 802.11b Wireless PCI Adapter
Operating System	Ubuntu Linux, Kernel version 2.6.12-9-386

Table 7.2: Node #2 Configuration

7.2 Experiment Environment

7.2.1 Node Configuration

A three node network, consisting of two desktop PCs and one laptop PC were used for the experiments. Configuration information for the PCs is described in tables 7.1, 7.2 and 7.3.

7.2.2 Wireless Environment

The wireless interfaces on each node were configured as follows:

```
IEEE 802.11-DS ESSID:"slottle"
Mode:Ad-Hoc Frequency:2.457 GHz Cell: 02:02:A5:2E:47:51
```

Type	Generic Desktop PC
IP Address	192.168.1.101
Processor	1.7 GHz AMD Athlon XP CPU
Memory	256Mb RAM, 40Gb HDD
Network Adapter	D-Link ACX 100 22Mbps PCI 802.11b+ Wireless Adapter
Operating System	Ubuntu Linux, Kernel version 2.6.17-10-generic

Table 7.3: Node #3 Configuration

```
Bit Rate=2 Mb/s Tx-Power=15 dBm Sensitivity:1/3
Retry limit:4 RTS thr:off Fragment thr:off
Power Management:off
```

The output above is derived from the **`iwconfig`** wireless configuration utility.

Nodes #1 and #3 were hidden to one-another, while Node #2 had full connectivity with the other nodes.

The nodes were in close geographic proximity to one-another, indoors within a single building. All nodes reported a good signal-to-noise ratio, thus communication errors were minimised.

The experiments were conducted in an environment free from interference from other 802.11 devices. No efforts were made to eliminate any other causes of interference. The experiments were conducted in the near proximity of devices such as a GSM cellular telephone, an RF wireless mouse and keyboard, and various domestic appliances.

7.3 Does the Amount of Time Required to Send a Constant Amount of IP Data remain Constant?

7.3.1 Introduction

The synchronisation algorithms described in section 5.5.1 assume that the amount of time required to transmit a constant amount of IP data remains constant over time. This experiment aims to test the validity of this assumption.

We anticipate that, occasionally, packet delivery will be delayed by factors beyond the control of our design. Such factors include lower-layer signalling and interference.

7.3.2 Hypothesis

While a small portion of packets may be delayed (and thus delivered with a high latency), over time the majority of packets will be delivered within a *relatively constant* amount of time. By relatively constant, we mean that the variance in delivery time is small relative to the mean delivery time.

7.3.3 Method

The **ping** program is able to time the round trip time of an IP packet.

ping uses the ICMP [31] protocol's mandatory ECHO_REQUEST datagram to elicit a response from a host or gateway. **ping** packets have an IP and ICMP header, followed by an internal structure and then an arbitrary number of "pad" bytes used to fill out the packet.

By default, **ping** transmits a small packet (56 bytes of Layer 3 data). This default is of a similar order to the size of a beacon, and is thus adequate for our needs.

The default interval of the **ping** program is one second, for repeated pings. This interval is sufficient to avoid the effects of 802.11's Diminishing Random Backoff associated with successive transmissions.

PC#1 was programmed to ping PC#2 repeatedly, at one second intervals, over a five minute period, using the following command line:

```
ping -c 300 192.168.1.98
```

The round-trip time is recorded for each ping. Because **ping** records round-trip times, it is actually timing two separate transmissions: the request and the response.

If the round-trip time is relatively constant, it is fair to assume that the one-way trip time is also relatively constant.

7.3.4 Results

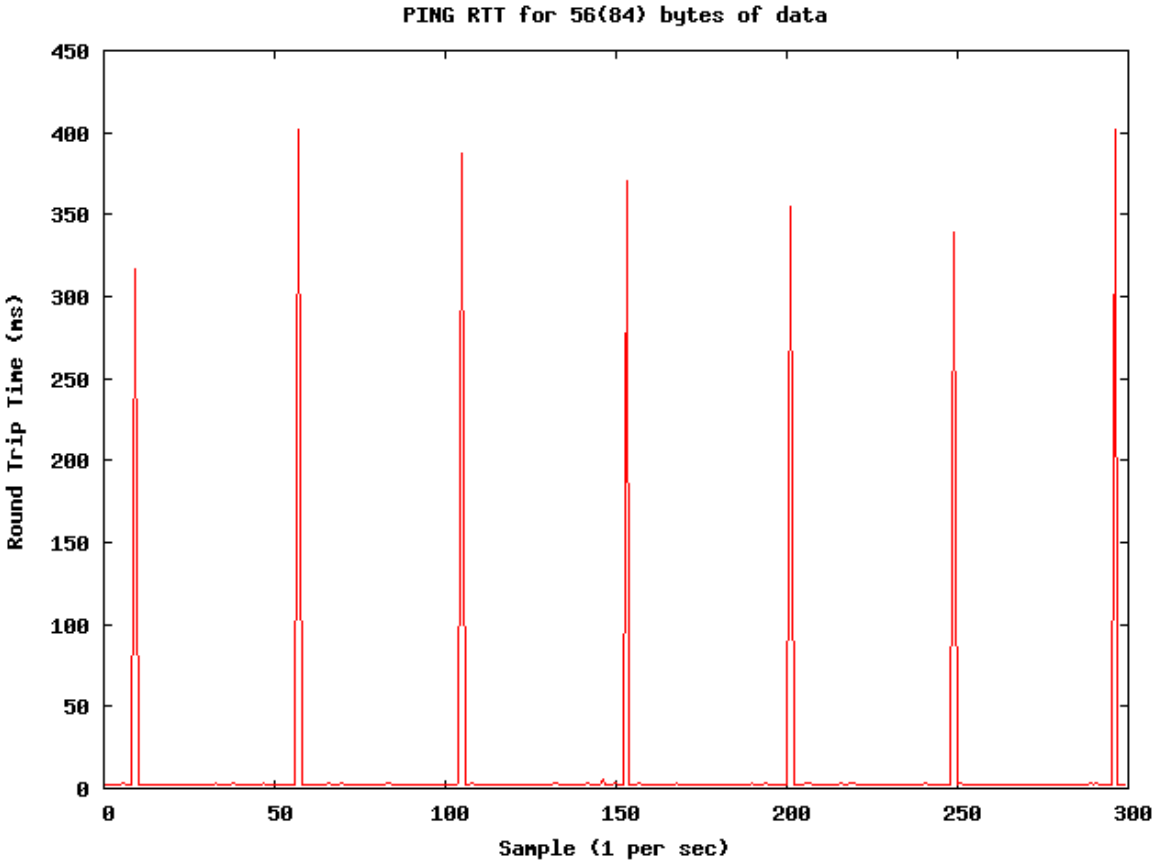
Figure 7.1 graphs the results of the experiment. We observe that most of the samples (97.7%) lie within a narrow range (between 2ms and 3ms), but that the graph has several outlying "spikes".

The spike samples occur approximately every 45th sample, and have a value greater than 300ms.

In order to investigate the cause of the spikes, the **ethereal** packet sniffer was used to capture and examine incoming and outgoing data on the wireless interface. The extract shown in table 7.4 shows that the spikes are caused by ARP[1] requests.

Figure 7.2 graphs the results with the "spike" samples removed. The samples now all fall between 2.3ms and 2.9ms. The mean delivery time is 2.56 ms with a variance of 0.03.

[1]The Address Resolution Protocol (ARP) operates at layer-3 and is used to resolve between layer-3 IP addresses and layer-2 MAC addresses.



ARP

Figure 7.1: Ping Round Trip Times

Source	Destination	Prot.	Description
192.168.1.100	192.168.1.98	ICMP	Echo (ping) request
192.168.1.98	192.168.1.100	ICMP	Echo (ping) reply
192.168.1.100	192.168.1.98	ICMP	ay
192.168.1.98	192.168.1.100	ICMP	Echo (ping) reply
D-Link_5c:08:b9	CompaqCo_2e:47:51	ARP	Who has 192.168.1.100?
CompaqCo_2e:47:51	D-Link_5c:08:b9	ARP	.100 at 00:02:a5:2e:47:51
192.168.1.100	192.168.1.98	ICMP	Echo (ping) request
192.168.1.98	192.168.1.100	ICMP	Echo (ping) reply

Table 7.4: Packet Sniffer Output

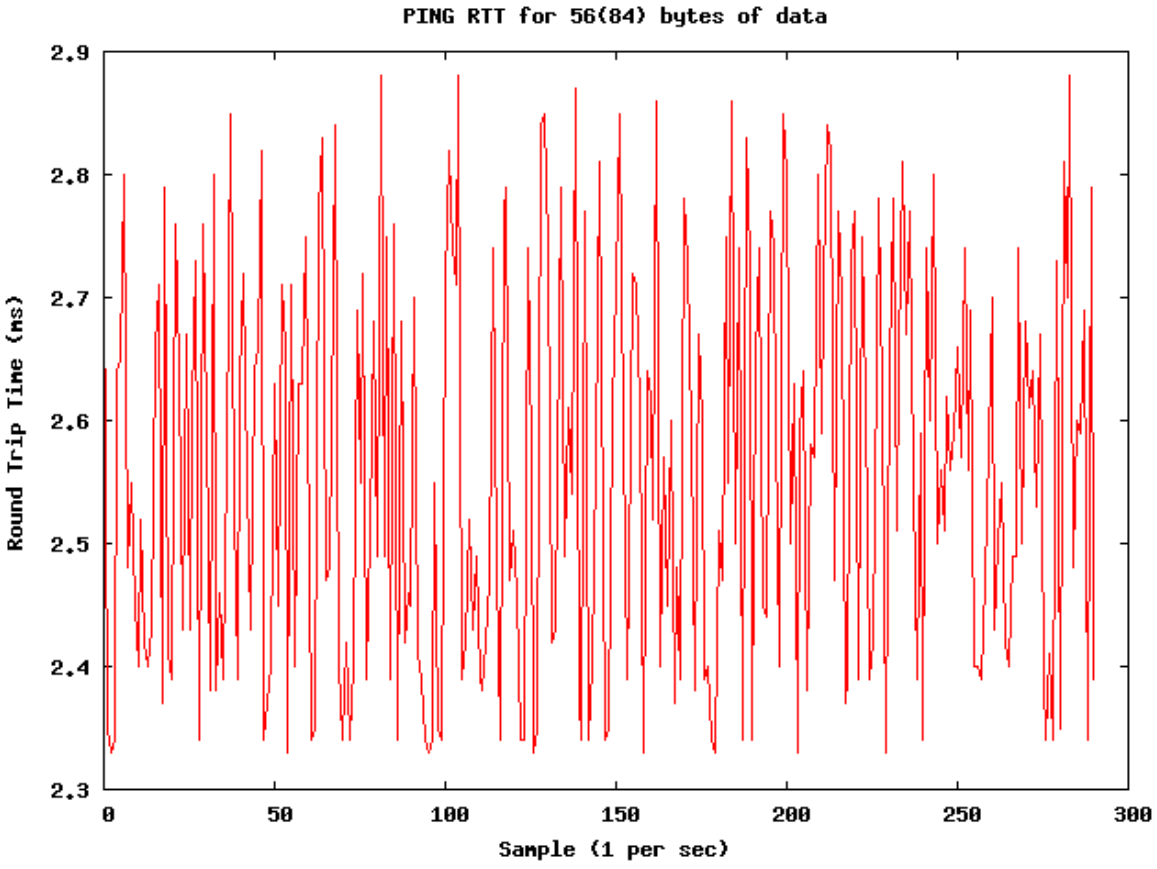


Figure 7.2: Ping Round Trip Times (Spikes Removed)

7.3.5 Summary

In our experiment, 97.7% of the samples had a mean delivery time of 2.56 ms with a variance of 0.03. This variance is very small, thus we conclude that the round trip time of an IP packet of constant size is relatively constant.

Occasional ARP requests can cause dramatic increases in beacon latency. Fortunately, they do not occur very often. Furthermore, it is possible to reconfigure the ARP cache refresh interval to reduce the frequency of ARP requests.

Broadcast transmissions do not require ARP resolution, because broadcasts are not addressed to any particular node. Because beacon packets are broadcast, the delays associated with the ARP protocol can be eliminated.

Packet aggregation requires all transmissions to be broadcast, and thus has the potential to eliminate the need for ARP requests.

In an environment with minimal interference, the delivery time of a constant amount of IP broadcast data may be said to be relatively constant.

7.4 Inter-packet Gap and Diminishing Random Back-off

7.4.1 Introduction

Section 4.2.2.1 discussed the inter-packet gap, which includes a random diminishing back-off period. The random element makes predictions of packet latency difficult.

The objective of this experiment is to discover the magnitude of the random element. Specifically, we wish to evaluate whether the random element is small or large, relative to the duration to transmit a packet.

If the random element is small, it will not have a very significant effect on the timing algorithms. A small random element can be compensated for by allowing extra time for any transmission, in order to avoid slot over-runs.

If the random element is large, this compensation approach becomes less feasible. The allowance of a large extra time would lead to excessive slot under-runs, resulting in inefficient bandwidth utilisation.

7.4.2 Method

The MTU[2] is a Layer 3 parameter which specifies the maximum size of an IP packet. Any packets which are larger than the MTU are fragmented into multiple smaller packets. The default MTU on most systems is 1500 octets.

The **ping** program allows for configuration of a specific packet size via the `-s` parameter.

A **ping** with packet size slightly less than the MTU (1450 bytes) was transmitted repeatedly over a five minute period, and the round trip time recorded. All spikes attributable to ARP were removed, as described in section 7.3.4.

The command line parameters used were:

```
ping -c 300 -s packetSize 192.168.1.98
```

The **ping** experiment was then repeated with a packet size slightly exceeding the MTU (1550 bytes).

Because **ping** records the round-trip time of a transmission, each sample will include the random element from the ping request as well as from the reply.

7.4.3 Results

Figure 7.3 graphs the recorded **ping** round trip times for 1450 bytes. The lowest measurement is approximately 11.9ms, and the highest measurement is approximately 0.8ms greater.

Table 7.5 is a packet-sniffer extract which shows that the 1550-byte ping was fragmented, as expected.

Figure 7.4 graphs the round trip times for 1550 bytes. The lowest measurement is approximately 260ms, and the highest measurement is approximately 160ms greater at 420ms.

[2]Maximum Transmission Unit

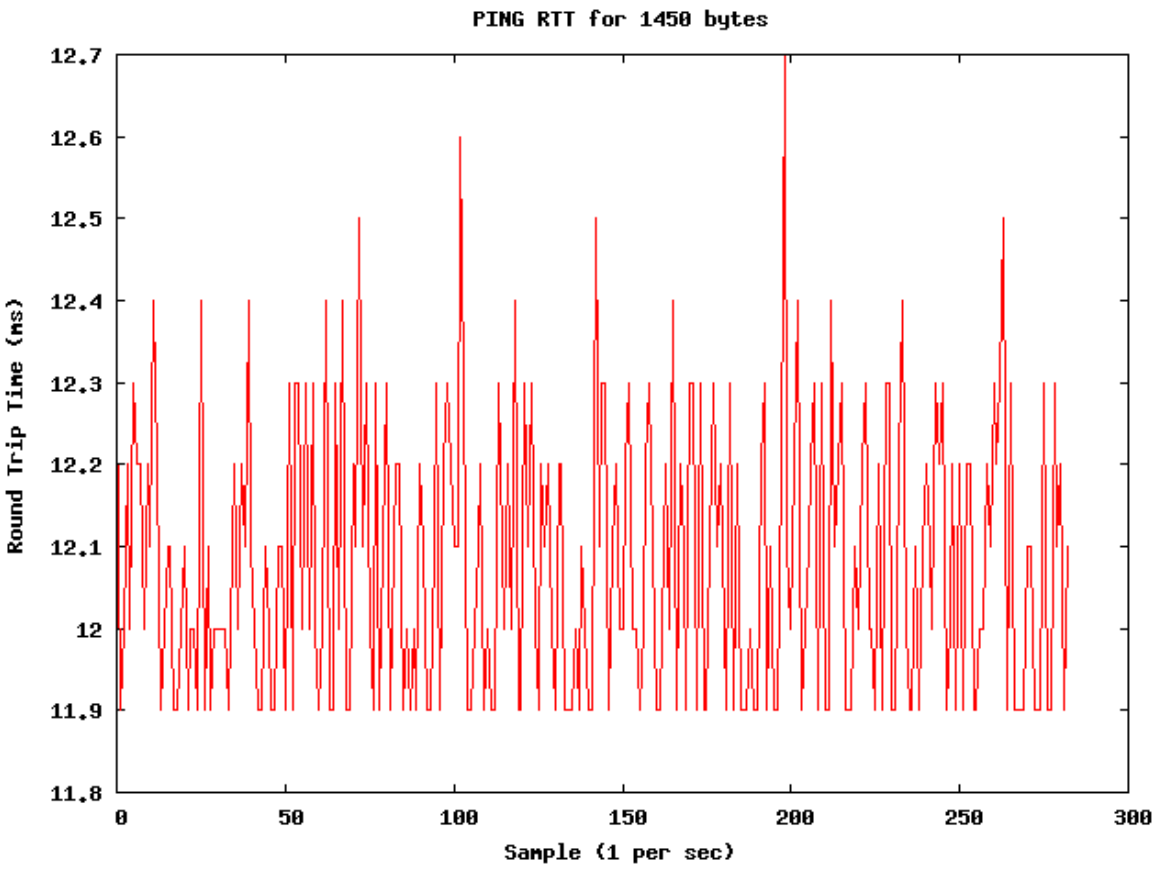


Figure 7.3: PING Round Trip Time for 1450 bytes

Source	Destination	Prot.	Description
192.168.1.100	192.168.1.98	IP	Fragmented IP protocol (proto=ICMP 0x01, off=0)
192.168.1.100	192.168.1.98	ICMP	Echo (ping) request
192.168.1.98	192.168.1.100	IP	Fragmented IP protocol (proto=ICMP 0x01, off=0)
192.168.1.98	192.168.1.100	ICMP	Echo (ping) reply

Table 7.5: Packet Sniffer Output for 1550 byte PING

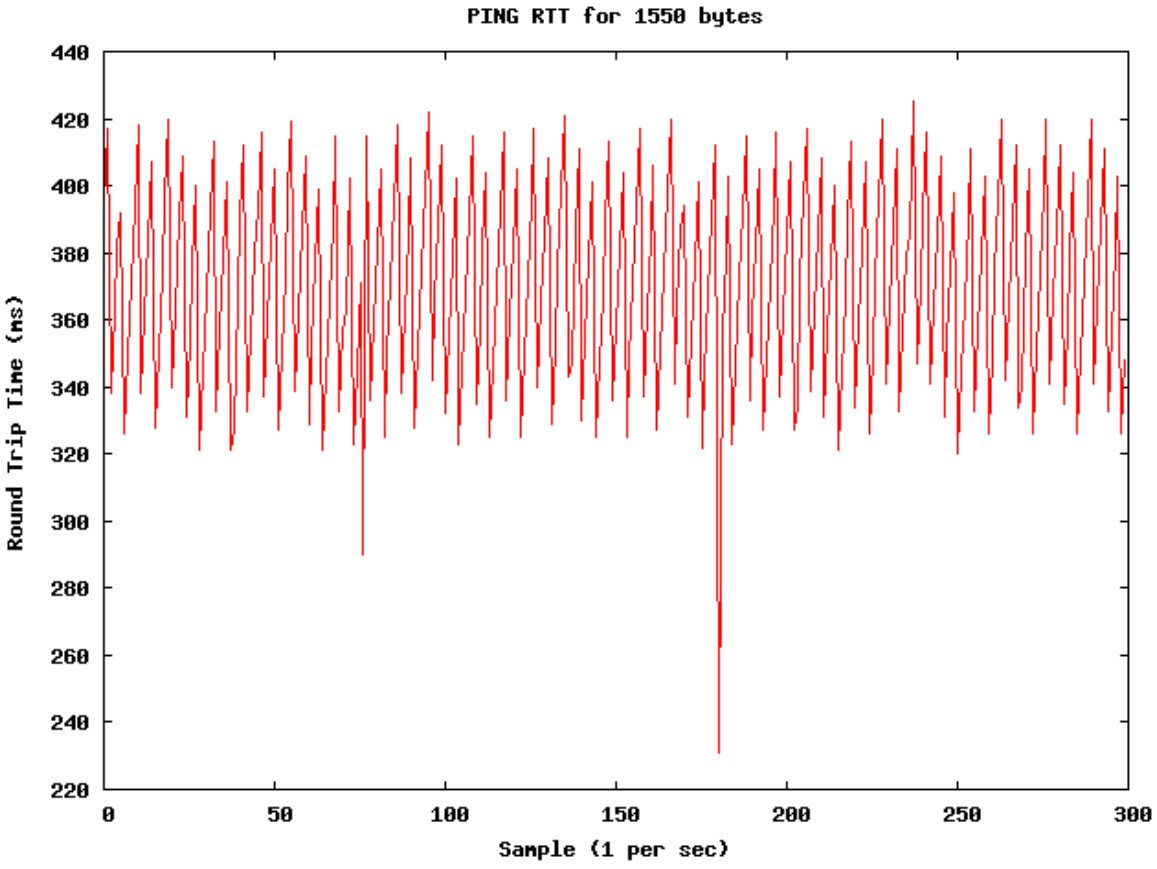


Figure 7.4: PING Round Trip Time for 1550 bytes

7.4.4 Discussion

The experiments for the unfragmented 1450 byte transmissions indicate that the range of the random element for a round trip is 0.8ms, thus the range of the random element for a single transmission is approximately 0.4ms.

The experiments for the fragmented 1550 byte transmissions indicate that the range of the random element for a round trip is 160ms, thus the range of the random element for a single transmission is approximately 80ms.

The size of the random element needs to be evaluated relative to the duration to transmit a packet. The lowest and highest measurements for a round trip were 260ms and 420ms respectively, thus for a single transmission we halve these values to 130ms and 210ms respectively.

80ms is large relative to either of these values.

7.4.5 Summary

The experiments show that the random element is large for the laboratory network. For this reason packet fragmentation should be avoided if possible, in order to optimise bandwidth utilisation.

Packet fragmentation can be avoided provided that all outgoing packets are smaller than the MTU. The size of the MTU can be adjusted in most network drivers.

Packet aggregation transmits a constant amount of data during each slot, and is thus able to guarantee that all outgoing packets are smaller than the MTU. When using FIFO queueing, there is no elegant mechanism to deal with packets bigger than the MTU.

Packet aggregation is thus preferred because it is able to avoid fragmentation.

7.5 Determining Beacon Latency

7.5.1 Introduction

Beacon latency was defined in section 5.5.2.3 as the difference in time between the moment when the sender transmits a beacon, and the moment when the receiver receives the beacon.

The objective of this experiment is to discover how beacon latency is affected by changes in packet size.

If the multiplexer uses packet aggregation, payload data is carried within the beacon packet. The size of the beacon packet is configured on each node at start-up.

In order to configure an appropriate packet size for a given slot duration, it is necessary to understand the relationship between beacon latency and packet size.

7.5.2 Method

The beacon packet size was varied and the effect on beacon latency was measured.

All nodes were configured with identical packet sizes throughout the experiment. For each packet size sampled, the application was run for a period of one minute and the mean beacon latency was recorded.

Beacons are transmitted as UDP broadcasts. Packet size is therefore specified in terms of UDP bytes transmitted.

The application reports beacon latency measurements in the output logs. These measurements are calculated using the formulae described in section 5.5.1.

The command line parameters used to start the application were:

```
sudo nice -n -20 ./slottle nodeID -l 100 -d 1000000 \
-a 500000 -y packetSize
```

The application was configured with a large slot duration (1 second) to avoid potential slot overruns, using the `-d` parameter.

7.5.3 Results

The results of the experiment are listed in table 7.6, and are graphed in figure 7.5.

Beacon Size (bytes)	Beacon Latency (ms)
128	2
256	3
512	4
1024	7
2048	12
4096	23
6000	33
8192	45
12000	66
16384	89
20000	109

Table 7.6: Beacon Latency Measurements

7.5.3.1 Predicting Beacon Latency

The graph in figure 7.5 can be seen to be almost perfectly linear. Given that the relationship between packet size and beacon latency is linear, we can predict the latter in terms of the former using a linear equation:

$$y = mx + c \tag{7.1}$$

If a straight line is fitted between the first and last samples on the graphs, values can be derived for m and c by solving simultaneous equations.

In the case of figure 7.5, solving the equations yields values of $c = 1.31078905$ and $m = 0.005384461$.

These values of m and c were derived for a specific node on our laboratory network. The values depend on the node configuration, and need to be recalculated for each node in a non-homogeneous network.

7.5.3.2 Effect of Packetisation

The maximum size of a transmitted packet is determined by a parameter called the MTU. The default MTU for most Linux 802.11 devices is 1500 bytes.

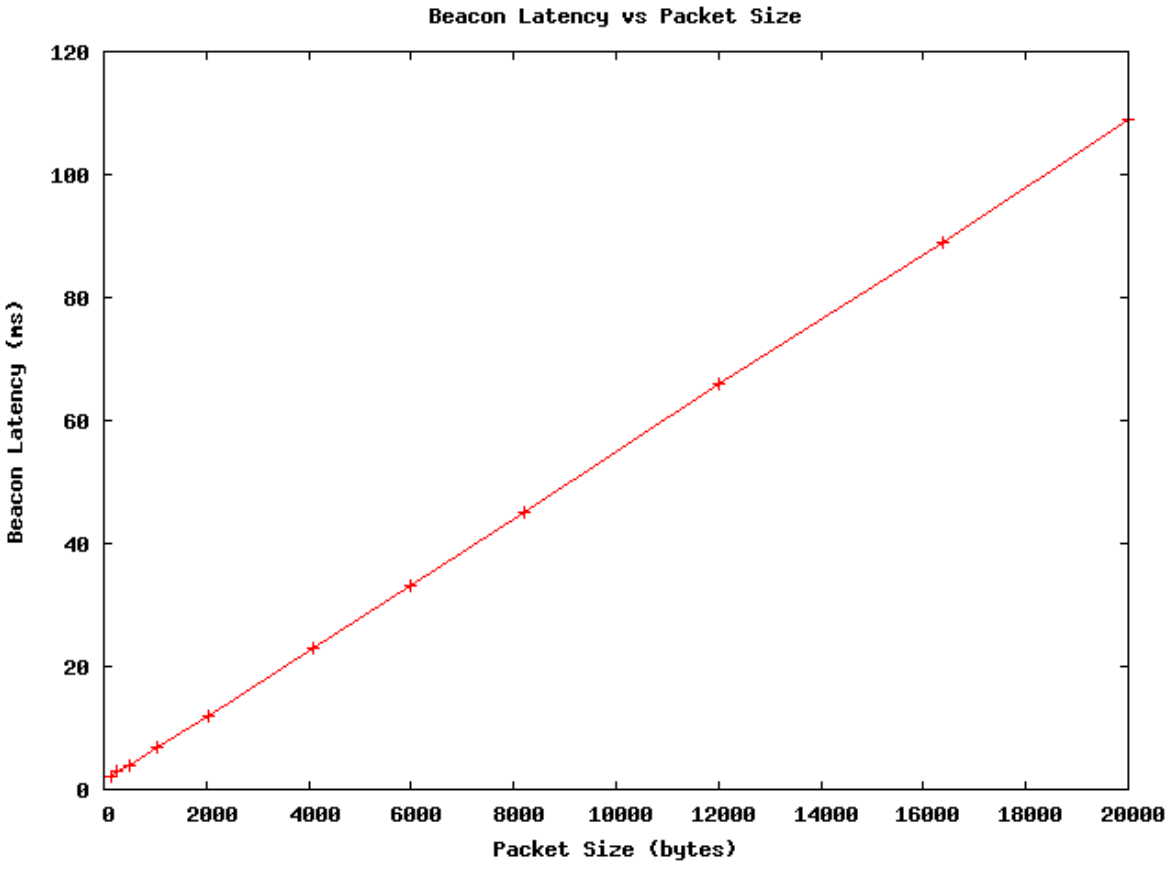


Figure 7.5: Graph of Beacon Latency Measurements

Any transmitted IP packet with a size exceeding the MTU is fragmented into multiple smaller packets. Each packet carries additional protocol overheads. Were the experiment to be repeated, and a much larger number of samples taken, it is anticipated that the graph would reveal a "stepped" shape. At each 1500-byte boundary, we predict a non-linear increase in latency as a result of the overhead associated with an additional IP packet.

The default MTU was not altered for this experiment. It is interesting to note that, in this experiment, the 802.11 diminishing random back-off is not invoked when the packet size exceeded the MTU.

This is because beacon packets are broadcast. In an 802.11 network, nodes reserve the medium for the duration of a broadcast. The diminishing random back-off, which is designed to judiciously share the medium, is not required when the medium is reserved.

The software application's ability to measure beacon latency can be used to test predictions. Should the mean beacon latency be found to be slightly less or more than desired, the payload size can be adjusted and the experiment repeated.

7.5.4 Summary

The relationship between beacon size and latency is linear. This relationship can be used to predict beacon latency, but predictions might be slightly inaccurate. The software application can measure beacon latency, and this facility can be used to check the accuracy of estimates.

Broadcast communications do not invoke the 802.11 diminishing random back-off when packets are fragmented. Packet aggregation ensures that all transmissions are broadcast, whereas FIFO queueing traffic is likely to be point-to-point.

7.6 Chapter Summary

The amount of time required to send a constant amount of IP data remains fairly constant for a single packet, thus predictions of transmission durations are feasible. Occasional ARP requests can cause delays, but these are only required for point-to-point and not for broadcast communication.

The range of the Diminishing Random Back-off period in the Inter-packet Gap is large, thus fragmentation of packets should be avoided if possible.

The measurements of beacon latency for nodes in our network show that the relationship between beacon size and latency is approximately linear for broadcast communications.

Packet aggregation is preferred to FIFO queueing because it uses broadcast communication and can avoid fragmentation.

Chapter 8

Discussion and Evaluation

This chapter seeks to evaluate the performance of the Layer 3 Reservation TDM scheme.

8.1 Introduction

The parameters which affect performance are introduced first, and their impact on performance is briefly discussed.

The first part of the evaluation assesses the effect that TDM parameters have on various network metrics.

The second part of the evaluation addresses the research questions introduced in section 1.4.

8.2 Introduction to the Parameters

The performance of the mechanism is dependent on numerous parameters. We categorise them as follows:

- TDM parameters: slot duration and the slot count.
- Layer-2 parameters, which affect the behaviour of the 802.11 MAC.

The sections which follow describe these in more detail.

8.2.1 TDM Parameters

The effects on performance of the two TDM parameters, slot count and slot duration, are discussed below.

8.2.1.1 Slot Count

Selection of an appropriate slot count value for a reservation TDM network involves a compromise:

- If node density is high and not enough slots are available, some nodes will be unable to obtain slots and will not be able to transmit at all.
- If a high a slot count is configured, nodes might have to spend a long time waiting for the arrival of their slots. This would result in an increase in average latency.
  A high slot count also reduces the amount of data that a node can transmit during each slot. This is because the slot table is transmitted at the start of each slot in the beacon, and the size of the slot table is directly related to the number of slots. When the slot table becomes larger, it takes longer to transmit and thus a smaller portion of the slot remains for payload use.

The slot count is designated c in the calculations which follow. The effects of c on various metrics are described later in this chapter.

8.2.1.2 Slot Duration

The slot duration measures an amount of time. During that amount of time, a finite amount of data can be sent. However, the entire slot is not available for payload data because an overhead of signalling data is transmitted at the beginning of each slot. Selection of the slot duration also involves compromise:

- A large slot duration will increase the average latency which nodes experience, but will allow higher throughput because the signalling overhead will be lower.
- A small slot duration will result in lower latencies, but less of the total bandwidth will be available for use because signalling overhead will be higher.

The slot duration is designated d. The effects of varying d are discussed in the sections which follow.

8.2.2 Layer 2 Parameters

Various layer-2 parameters which are configurable on 802.11 devices affect performance. This section introduces these parameters.

8.2.2.1 Raw Data Rate

802.11 supports multiple data rates. An 802.11b device supports up to 11 Mb/s, but can also transmit at lower speeds such as 5.5 Mb/s, 2 Mb/s or 1 Mb/s. Newer 802.11g devices support up to 54 Mb/s.

The lower data rates allow for communication under less than ideal conditions. An 802.11 network interface card (NIC) can be configured either to a fixed data rate, or to automatically select a data rate dependent on conditions which it senses in the environment.

The TDM scheme offers an equal slice of time during each slot. The amount of data that can be sent during that slot is clearly dependent on the raw data rate.

Because 802.11 uses a shared medium, only a portion of the bandwidth is normally available to each node. The TDM scheme, however, attempts to ensure that the medium is dedicated to just one node for the duration of a timeslice.

The overheads imposed by the 802.11 MAC and by higher network levels mean that significantly less than the full bandwidth is available to any node for payload utilisation.

8.2.2.2 Fragmentation Threshold

Fragmentation allows an 802.11 device to split an IP packet into smaller fragments. This might improve performance in noisy environments. The fragmentation threshold is a configurable parameter which allows one to specify the maximum size of any fragment.

Fragmentation adds overhead and reduces available bandwidth.

Our timing routines assume that there is a linear relationship between packet size and the duration required to transmit that packet. Packets which are fragmented will take longer to transmit than expected, and will thus tend to cause slot over-runs.

Fragmentation must be disabled in order for the application to function correctly.

8.2.2.3 RTS Threshold

Most NICs allow configuration of a packet size threshold, beyond which the RTS/CTS mechanism will be activated. RTS/CTS is designed to alleviate hidden node problems, but the mechanism adds latency and protocol overhead.

RTS/CTS must be disabled in order for the application to operate correctly.

The RTS mechanism can be disabled by configuring a threshold value equal to the maximum packet size.

8.2.2.4 Retry Limit

Most NICs support MAC retransmissions. This allows for retransmission of corrupted data. The retry limit allows for configuration of maximum number of retries, as well as for the packet lifetime. However, if Layer 2 retransmits, a given packet will take longer than expected to send.

Retries can thus lead to slot over-runs. The higher the retry limit, the more likely it is that a data error will cause an over-run. We thus suggest that the retry parameter be configured to its minimum value.

Retries are the result of communication errors. Retries can be minimised through good network design.

8.2.2.5 Summary

Various layer-2 parameters can affect performance. The TDM scheme requires that RTS/CTS and fragmentation be disabled. Retries can lead to over-runs in noisy environments.

8.3 Deriving Performance Metrics

This section assesses the performance of the reservation TDM scheme in terms of the parameters introduced earlier.

8.3.1 Signalling Overhead

Under the reservation scheme, nodes which have reserved slots are required to transmit a beacon at the start of any slots they own.

We define signalling overhead as that portion of a node's time slot which is consumed in transmitting the beacon. Signalling overhead is expressed as a ratio.

The size of the beacon packet is constant, and it is assumed that a beacon will take a constant amount of time to transmit, *ceteris paribus*. The duration of time required to transmit a beacon is known as the beacon latency. Beacon latency measures the time difference between when a node begins transmitting a beacon, and when it is received by other nodes.

The application is required to measure beacon latency, as described in section 5.5.1. Let l_b equal the beacon latency of a node. Then the signalling overhead o can be measured in terms of the slot duration d :

$$o = \frac{l_b}{d} \tag{8.1}$$

Because the signalling overhead is constant, if one reduces the slot duration the percentage of the slot which is utilised for signalling increases. This means that a reduction in slot duration leads to a smaller percentage of total bandwidth being available to each node.

By increasing the slot duration, the portion of bandwidth used for signalling decreases. Each node thus has a greater percentage of the total bandwidth available. But an increased slot duration leads to increased latency, and this might be undesirable, depending on for what purpose the network is predominantly used.

8.3.2 Payload Capacity

Available capacity is defined as that portion of a node's time slot which remains after the beacon has been transmitted. The payload capacity is calculated as follows:

$$p = d - l_b \tag{8.2}$$

8.3.3 Latency

The latency l experienced by a node for payload transmissions is defined as the difference in time between the moment when a node commences a transmission, and the moment when the transmission is received in full by other nodes.

Our latency calculations do not allow for the contention delays when a node attempts to acquire a slot. Because all nodes attempt to reserve a single slot immediately after start-up, we assume that the node has already acquired a slot.

The latency experienced by a node which has reserved a slot may be subdivided into TDM, Beacon and Channel latencies. The three types of latency are described below:

8.3.3.1 TDM Latency

TDM latency is a result of the TDM scheme. The TDM-imposed latency is the duration that a node spends waiting for the commencement of its slot. We denote TDM latency l_t.

The l_t associated with a given transmission depends on how long a node has to wait for the arrival of its slot. In the best case, the node wishes to transmit payload data immediately before it completes transmission of the beacon. In this case, there is no TDM latency. Thus:

$$l_t = 0 \tag{8.3}$$

The worst case occurs immediately after this, when the beacon has already been sent, and MUX has detected no queued data to transmit. In this case, the node has to wait the full duration of c slots.

$$l_t = c * d \tag{8.4}$$

The distribution is uniform, thus the mean TDM latency experienced by a node is:

$$l_t = \frac{c * d}{2} \tag{8.5}$$

8.3.3.2 Beacon Latency

Beacon latency is that portion of latency associated with transmission of the beacon packet. A beacon packet precedes any payload transmission under the reservation TDM scheme. Beacon latency l_b is assumed to be constant for a given node.

8.3.3.3 Channel Latency

We define the channel latency l_c as that portion of latency which is attributable to layers 3 and below in the network stack.

Channel latency is related to the amount of data to be transmitted, and to the layer-2 data rate.

Channel latency is also affected by CSMA/CA's carrier sensing mechanisms.

A payload packet is always immediately preceded either by a beacon packet, or by another payload packet. Therefore all payload packets invoke the diminishing random backoff described in section 2.6.3.

Channel latency thus has a variance which is related to the diminishing random backoff period.

8.3.3.4 Summing the Latencies

The latency experienced by a node is equal to the sum of the three components:

$$l = l_t + l_b + l_c \tag{8.6}$$

But l_t can vary between 0 and $c * d$, depending on how long the node has to wait for its slot to arrive.

The minimum latency, when $l_t = 0$, is calculated as:

$$l = l_b + l_c \tag{8.7}$$

The maximum latency, when $l_t = c * d$, is calculated as:

$$l = c * d + l_b + l_c \tag{8.8}$$

Finally, the mean latency is:

$$l = \frac{c * d}{2} + l_b + l_c \tag{8.9}$$

8.3.3.5 Summary

The latency experienced by an individual node depends predominantly on how long that node has to wait for the start of its next slot.

In the best case, the node does not need to wait for its slot to arrive, and the latency is constant.

The mean latency is directly related to the product of slot count and slot duration.

An upper bound for latency can also be calculated.

8.3.4 Layer-3 Throughput

Layer-3 throughput is indirectly related to payload capacity.

An experimental analysis of the actual throughput achieved by the prototype application would not give an accurate indication of the potential of the TDM mechanism, because the prototype MUX supports only inefficient FIFO queueing.

Our analysis thus focusses on the theoretical throughput achievable through packet aggregation. We assume that each node reserves a single slot.

In the case of packet aggregation, IP data would be encapsulated within packets of predetermined size *s* octets. We assume that no additional header information is required for each IP packet, which is reasonable because successive IP packets in a data stream can be identified by their IP headers. each of which specifies the size of the respective packet and thus implicitly the offset of the following packet.

If we assume that each aggregated packet has a header of size *h* octets, which contains the beacon fields and any other required information, then the amount of data *p* available for payload utilisation during each slot is:

$$p = s - h \tag{8.10}$$

If *d* is the slot duration, and *c* is the slot count, then the node can transmit *p* octets every duration *c*d*. The maximum achievable layer 3 throughput *t* is thus:

$$t = \frac{p}{c * d} \tag{8.11}$$

8.3.5 Scalability

Were an additional node to join a cluster, there might be insufficient slots available to fill the needs of all nodes. The scalability of TDM protocols in general is thus not as good as that of contention-based protocols.

The reservation protocol propagates information for exactly two hops, in order to forward knowledge of hidden nodes. If nodes reserve exactly one slot each, then the number of nodes within a two cluster is limited to the configured slot count c.

Because there are latency and throughput penalties associated with an increased slot count, the network designer needs to select this parameter carefully.

It is important to note that the value of c affects the maximum *density* of nodes, not the total number of nodes which can be accommodated on the entire network. Nodes separated by more than two hops are able to independently re-use slots.

If insufficient free slots are available to satisfy the needs of a dense cluster of nodes, various strategies could be employed:

8.3.5.1 Slot Count Reconfiguration

The network slot count could be increased by manual intervention. An increased slot count would result in increased latency and reduced throughput for all nodes.

This would require reconfiguring and restarting of the entire network, and is clearly not desirable.

8.3.5.2 Spatial Reuse through Transmit Power Reduction

An alternative strategy would be to reduce the transmit power of nodes in the cluster as suggested in [40], thereby reducing the cluster size. Because 802.11 is a shared medium, the higher the number of nodes in a cluster, the less the bandwidth available to each node. Thus a reduction of

node density via a reduction of transmit power would lead to improved throughput for individual nodes.

8.3.5.3 Channel-Partitioning the Network

802.11's physical layer supports multiple channels. Effective node density can be reduced by configuring some nodes some nodes in the network to use a different channel.

This approach would require some nodes to be fitted with dual network interfaces, in order to route between the different channels.

8.3.5.4 Directional Antennae

Nodes in an ad-hoc network are most commonly fitted with omni directional antennae, which transmit a signal in multiple directions. This is appropriate for a network in which the physical topology is unknown.

Should the physical topology be known in advance to the network designer, it is possible to employ directional antenna, such as Yagi arrays. These can be pointed in specific direction, and are commonly used for point-to-point connections.

By excluding the signals of adjacent nodes, directional antennae are able to reduce the density of nodes within a cluster.

8.3.5.5 Multiple Slot Reservations can Mitigate Latency Penalties

Somewhat paradoxically, if nodes were allowed to reserve multiple slots they might be able to arrange their slots so as to mitigate the increased latency effects of a high slot count. For example, a node could reserve two slots equidistant from one-another, thus halving the average latency associated with waiting for slot arrival.

Were nodes allowed to reserve multiple slots rather than just one, a relatively higher slot count would be required.

8.3.5.6 Summary

TDM protocols may not scale as well as contention based protocols, because the slot count c limits the number of nodes which may be present in any two-hop cluster.

It may not be desirable to increase c, because this would lead to performance degradation. We thus proposed various alternative strategies which could be employed to mitigate the scalability limit.

Because the value of c affects the maximum *density* of nodes, not the total number of nodes which can be accommodated on the entire network, a well-designed network will be scalable.

8.3.6 Summary

This section evaluated how metrics such as signalling overhead, latency, throughput and scalability are affected by TDM parameters.

8.4 Research Evaluation Questions Revisited

8.4.1 What are the benefits of the mechanism?

The primary benefit of this mechanism is its applicability to existing hardware.

This means that the mechanism has the potential to be applied to existing networks where nodes suffer from bandwidth starvation or hidden-node collisions, as a remedial measure.

Because it is implemented as a user-space application rather than as a kernel modification, the mechanism is very quick and easy to deploy.

The mechanism facilitates the building of low-cost mesh networks which can span great distances. It may be of particular interest in areas where little other telecommunications infrastructure exists, such as rural communities.

TDM offers the potential to support real-time media such as IP telephony and video streaming, supported by QoS[1] protocols.

8.4.2 What complications and issues arise?

8.4.2.1 Difficulties Relating to the 802.11 MAC

The CSMA MAC used by 802.11 imposes various difficulties. These include:

[1] Quality of Service

The Exposed Node Problem, discussed in section 2.9, prevents parallel point-to-point transmissions where such transmissions could potentially take place without either receiver experiencing a collision.

Carrier Sensing, as performed by the 802.11 MAC prior to any transmission, is not required under the TDM scheme. The carrier sensing function results in unutilised bandwidth.

Random Back-off 802.11 imposes a random back-off period after a successful packet transmission, in order to allow other nodes a chance to transmit. Because the Reservation TDM scheme is able to guarantee a node exclusive use of the medium for the slot duration, this back-off mechanism is not required and only results in unutilised bandwidth. Worse still, the random element of the back-off period makes accurate timing of packet transmissions difficult.

Fragmentation and Retransmission Limits are configurable parameters on most 802.11 interfaces. Fragmentation and retransmission can wreak havoc with the timing algorithms described in section 5.5.1. These mechanisms, although desirable in some circumstances, need to be disabled in order for the application to function correctly.

8.4.2.2 Implementation Issues

A modern operating system typically is able to multi-task, executing a number of processes in parallel. It does through time-slicing, under which each process is granted a small slice of CPU time [44].

The need for accurate timing in the TDM scheme poses difficulties on a non-real-time multi-tasking operating system with a shared processor. There is no guarantee that the process will be executing on the CPU at the moment when an operation needs to take place.

Most of the implementation issues, described in chapter 6, focus on methods of achieving accurate timing under a non-real-time multi-tasking operating system.

8.4.3 What limitations can be identified and quantified?

8.4.3.1 Overheads and Configuration Tradeoffs

The Layer-3 Reservation TDM scheme imposes overheads in terms of bandwidth consumption and increased latency. The overheads depend on the configuration of certain parameters.

The selection of critical configuration parameters such as the slot duration and slot count impose limits. The slot duration must be configured to balance demands for low latency versus the need to optimise throughput. The slot count must be high enough to support the node density, but a high slot count increases latency and overheads.

8.4.3.2 Size of payload

It is difficult to calculate how much data can be transmitted during a given slot duration. An over-run is a bad occurrence in a network, and should be avoided. Under-runs lead to decreased network utilisation. It is better to be conservative, and to allow for idle time toward the end of the slot to avoid potential over-runs.

The result is a loss of useable bandwidth.

8.4.3.3 Minimum Slot Duration

There is a limit to how small a slot duration can be configured. A finite amount of time is taken in any network to transmit the beacon packet which the TDM scheme requires. A slot duration equal to or less than that amount of time would yield a network unable to carry any payload, thus useless.

8.4.3.4 Sub-Optimal Bandwidth Utilisation

Because the proposed solution operates at the IP layer, our granularity is that of a single IP packet.

A single slot has a constant duration. If the remaining slot duration is insufficient to send the next packet, no data can be transmitted. The remaining slot duration is wasted, resulting in sub-optimal bandwidth utilisation.

8.4.3.5 Scalability

When the number of nodes within a cluster changes, it is not easy for a TDM protocol to dynamically change its frame length and time slot assignment. Thus the scalability of a TDM protocol may not be as good as that of a contention-based protocol.

8.4.3.6 Buffering Difficulties with Access Points

As an alternative to a wireless network card, the application was also tested using a standard 802.11b access point, connected to the host via an Ethernet cable. Under such a configuration, the timing algorithms yielded erratic results and thus the application did not perform well. We attribute the erratic results to buffering of outgoing data by the access point, which did not send packets immediately.

It would thus appear that the application requires the host to have an on-board wireless interface, rather than using an external device such as an access point.

Wireless access points have largely been superceded by wireless routers, which offer enhanced functionality at a comparable price. Because each node in an ad-hoc network is a router, wireless routers are better suited than access points for building such networks. Many wireless routers are Linux-based, or can be upgraded to a Linux-based distribution such as OpenWRT. Such routers offer an ideal platform for the deployment of the application.

8.4.4 What are the costs, in terms of latency and bandwidth, of the mechanism?

The performance penalties of L3 TDM depend on the configuration parameters selected by the network designer, as well as on the nature and activity of the network usage, and the needs of the users. The effect of configuration parameters on network performance was discussed earlier in this chapter.

Users of some networks, such as those with few nodes and little traffic, will find that L3 TDM introduces unacceptable latencies and throttles their bandwidth. Layer 3 TDM might not be applicable to these networks.

In other networks, such as community networks, users might find that hidden node problems render the network unusable at busy times. Distant nodes might find that they are starved of bandwidth. In such cases, the performance penalty of L3 TDM will be a small price to pay for the benefit of having a useable network.

8.4.5 Does the mechanism promise a useful solution?

The limitations of Layer 3 TDM have been discussed in depth in this dissertation. What becomes clear is that, like any technology, Layer 3 TDM would be applicable and of benefit only in certain circumstances.

The bandwidth and latency penalties may render the technology unsuitable for some high-bandwidth or low-latency applications, such as computer gaming or real-time media. For other applications, such as e-mail or World Wide Web browsing, the bandwidth and latency penalties would be acceptable. Because of the low cost of this technology, and its compatibility with existing and even outdated equipment, it might be particularly applicable to 3rd world, rural or previously disadvantaged communities.

8.5 Summary

This chapter evaluated the performance of the Layer 3 Reservation TDM scheme.

The parameters which affect performance were introduced. The first part of the evaluation assessed the effect that critical parameters had on various network metrics. The second part of the evaluation addressed the research questions introduced in section 1.4.

Chapter 9

Conclusion & Future Work

This chapter concludes the dissertation. The research objectives and questions are reviewed, and the specific contributions made by this work are discussed.

9.1 Introduction

This research set out to address two problems which affect 802.11 ad-hoc networks, via the design of a software mechanism applicable to existing devices:

- Overcome the Hidden Node Problem
- Provide fair bandwidth sharing among nodes

Drawing on the work of other researchers, we discovered MAC algorithms for TDM networks which could offer the key to our objectives. In order for our solution to be applicable to a wide range of existing devices, we approached the problem at Layer-3. Algorithms were developed and implemented to provide distributed local synchronisation, in order to facilitate virtual TDM. A simplified, distributed TDM-MAC algorithm was implemented over the virtual TDM to achieve the project objectives. These are discussed further below.

9.2 Correctness of Solution

This section seeks to confirm the correctness of our solution in fulfilling the research objectives.

9.2.1 Overcome the Hidden Node Problem

Hidden node problems have been shown to cripple the performance of some networks. The problem was tackled by dividing time into slots, and requiring nodes to reserve those slots prior to using them. A distributed slot reservation algorithm ensures that hidden nodes are identified, and collisions are avoided. We implemented a version of RR-ALOHA to overcome the hidden node problem.

In [27], Borgonovo *et al* demonstrate the correctness of RR-ALOHA in preventing hidden node collisions. We see no merit in reproducing their work.

We thus assert that RR-ALOHA correctly meets our first research objective.

9.2.2 Provide Fair Bandwidth Sharing among Nodes

CSMA/CA, used in 802.11, is asynchronous and does not provide fair bandwidth sharing among nodes. By implementing virtual TDM, we divided time into slots. TDM provides the basis for fair bandwidth sharing among nodes, as every node is guaranteed a fair slice of time during a reserved slot.

Heusee *et al* [39] assess the fairness objective in 802.11 networks, and conclude that Time Division fulfils it best.

Based on this, we assert that TDM correctly meets our second research objective.

9.2.3 Applicable to Existing Devices

Our scheme does not require any modification to the MAC layer, nor to any existing protocol. It is implemented as a user-space Linux application.

This makes the application easy to deploy on computers with existing wireless interfaces. It can also be deployed on a large range of wireless routers and access points, under OpenWRT (section 2.3). The application could be ported to other platforms without great difficulty.

The primary benefit of implementing TDM at Layer 3 is the immediate availability of the solution. Our solution is applicable to existing networks which suffer from the problems this research addresses.

The final research objective is thus correctly met.

9.3 Summary of Work Covered

The specific contributions which this research has made are discussed in this section.

9.3.1 Concept

We believe that the concept of Reservation TDM at Layer 3 is original. This research draws heavily on the work of others, but combines that existing research in a novel way. We are not aware of any other research having been conducted in this specific area.

9.3.2 A Potential Solution to a Real Problem

Ad-hoc networks built with 802.11 equipment today suffer from hidden node problems and are unable to guarantee each node a fair share of bandwidth. There is presently no solution to this problem that is applicable to existing devices. Our research offers the potential of an applicable, workable solution.

9.3.3 Feasibility Findings

Our prototype implementation of the layer 3 scheme demonstrates that it is feasible. Our evaluation of the scheme quantifies its performance.

The results of this study showed that Layer 3 TDM is feasible, and highlighted the constraints of the scheme.

The FIFO Queueing approach, as used by Frottle, does not suit Layer 3 TDM well because it leads to non-deterministic timing and inefficiency. Packet aggregation is the preferred approach.

9.3.4 Designs

Various software designs were required and produced in order to implement the Layer 3 TDM scheme. The most interesting of these are discussed below:

9.3.4.1 Application Design

Chapter 5 presents a design for an application which implements a virtual MAC layer. Our design draws heavily on other research in related areas to fulfil the unique requirements of this research. A prototype application was developed to both test and demonstrate the design.

9.3.4.2 Synchronisation Routines

This document describes a synchronisation scheme (section 5.5.1) which is suitable for ad-hoc networks. The scheme leads to loose synchronisation of the clocks of nodes with those of their neighbouring nodes, and does not depend on any central or real clock. It operates in a completely distributed manner.

9.3.4.3 Novel Latency Measurement Algorithms

The synchronisation routines depend on a knowledge of the latency associated with a beacon transmission. A node needs to know how long ago another transmitted a timestamp, in order to correct the timestamp once it is received.

An algorithm, based on that used by NTP, enables two nodes to estimate their respective latencies by halving the round-trip-time latency of a packet.

A new algorithm is proposed, which enables three interconnected nodes to actually measure one-another's latency, providing better accuracy than the round-trip-time method.

9.3.5 Software Application

During the course of this research, we have developed a functional prototype software application, which has the potential to be developed into a fully functional application.

The software application enables us to thoroughly test and refine our design, and demonstrates the feasibility of this research.

9.3.6 Performance Evaluation & Parameterisation

The performance of a laboratory network running the prototype application was studied for various performance metrics. The results were analysed and presented graphically.

By combining the results of the performance measurements with an analysis of the IP and 802.11 MAC network layers, we were able to derive models which can be extrapolated to predict performance under various conditions.

The predictions of these models were compared to data captured from a real network in the laboratory to prove their validity.

9.4 Future Work

9.4.1 Payload Optimisation

Section 4.2.3 suggests various ways in which the slot utilisation can be optimised. At present, the prototype application accepts static parameters of maximum packet count and byte count per slot.

We believe that the most efficient means of utilising the slot is to use a type of packet aggregation. In this way, multiple small packets can be combined into one or more large packets. The packets can be deaggregated by the receiving nodes.

Packet aggregation would minimise the Layer 2 and Layer 3 protocol overheads, thus improving throughput. If the entire payload for a slot could be sent in a single packet, it would also eliminate the inter-packet gap imposed by the 802.11 MAC, thus further improving throughput.

Because the inter-packet is followed by a random back-off, more accurate timing is possible for the first packet transmitted during a slot than for subsequent packets. With more accurate timing, the application could make more optimal use of time slots.

A disadvantage of packet aggregation is that it would require additional processing by all nodes in the network.

9.4.2 Enhanced slot reservation scheme, allowing multiple slot reservations

The prototype application implements a simplified slot reservation scheme, which allows each node to reserve a single slot. There are instances when it might be desirable for a node to reserve multiple slots.

A difficulty of allowing nodes to reserve multiple slots is that a distributed algorithm needs to be found to ensure a fair allocation of slots. Without such a mechanism to ensure fairness, some nodes might starve others of bandwidth.

9.4.3 Adjustable Transmit Power

The use of TDM imposes some limitations. One of these is that the slot count is statically configured. If node density in a network is high, there might be inadequate slots available to meet the needs of nodes.

One way to remedy this problem is to use power control, as described in [28]. If one reduces the transmit power of all nodes in a cluster, it has the effect of reducing node density. This is also desirable because 802.11 utilises a shared medium, and if too many nodes are active the bandwidth becomes scarce.

9.4.4 Implement Quality of Service

The CCITT Recommendation E.800 defines QoS as the collective effect of service performance which determines the degree of satisfaction of a user of the service.

TDM guarantees each node a repeating time slice of fixed duration. This allows each node to send a fixed amount of data during each slot. Because TDM offers each node a bandwidth guarantee, it can facilitate QoS via services such as IntServ[5] or DiffServ [4].

9.4.5 Optimisation through Integration with Routing Protocols

A longer term study of the interaction of routing protocols with the Layer-3 TDM mechanisms might yield further potential optimisations. For example:

- If the routing protocol could know how large the outgoing queue of a node is, or what the node's recent slot utilisation has been, it could choose to route data via those nodes which have little to transmit. This would optimise throughput for all nodes in the network.
- The slot reservation scheme could be optimised to grant additional slots to nodes forwarding data on behalf of other nodes.

- The TDM scheme adds latency because nodes have to wait for their slots to start before transmitting. If the routing protocol had knowledge of slot allocations, it could select a route so as to minimise this latency.

Further research in this area could certainly be of interest. Many of the issues described above apply as well to conventional Layer-2 TDM as they do to Layer-3 TDM. This means that the research could draw on and contribute to existing bodies of knowledge.

9.5 Concluding Remarks

This research set out to address real problems in 802.11 ad-hoc networks. It proposes a solution which is applicable to existing devices. A prototype application has been developed to demonstrate the feasibility of the solution.

We hope that the contributions made by our work will be of value to future researchers.

Further work will naturally be required if this solution is to be deployed on real networks. The potential benefits of the solution undoubtedly warrant the work.

If the **`slottle`** application is ever developed and used as widely as the **`frottle`** application which inspired it, we will be entirely satisfied that our research will have made a worthy contribution. Toward this end, and in acknowledgement of the work of the many others on which ours draws, we open-source the **`slottle`** source code under the GNU GPL.

Appendix A

Glossary of Acronyms

ADSL Asynchronous Dial-up Subscriber Line

CSMA/CA Carrier-Sense Multiple Access with Collision Avoidance

CSMA/CD Carrier-Sense Multiple Access with Collision Detection

DCF Distributed Co-ordination Function of 802.11

FIFO First-In-First-Out, refers to queueing

GPL General Public Licence, from the GNU foundation.

GPS Global Positioning System, which provides an accurate reference clock

MAC Medium Access Control layer of the OSI reference stack.

MANET Mobile Ad-Hoc Network

MUX Multiplexer

NIC Network Interface Card

NTP Network Time Protocol

PCF Point Co-ordination Function of 802.11

QoS Quality of Service, facilitating bandwidth reservations

TDM Time Division Multiplexing

WRT54G A model of wireless router supplied by LinkSys.

VoIP Voice over the Internet Protocol, for telephony.

Appendix B

Using the Prototype Application

B.1 Introduction

This appendix describes the configuration, installation and startup requirements of the prototype application, called **`slottle`**.

A significant part of the instructions in this section are based on the instructions distributed with Frottle [15].

B.2 Configuration

B.2.1 Configuring a NIC to ad-hoc mode

A shell script is provided, called SETUPADHOC.SH. Edit the script as follows:

- Substitute the name of your network interface for eth1
- Substitute a unique IP address for the one specified. For convenience, we configure all nodes to the class-C 192.168.1.0 network, which is unused on the Internet. We select a unique number for each node in the range 2..254, and use that number for both the NodeNumber and the IP address. eg, node 123 has address 192.168.1.123.

If the your NIC does not disable them by default, disable RTS/CTS and Fragmentation.

The broadcast address for the 192.168.1.0 class-C network is 192.168.1.255, which is the default broadcast address used by the application.

Now run the script. You will require root privileges:

```
sudo ./setupadhoc.sh
```

The contents of the script are shown below:

```
iwconfig eth1 mode ad-hoc essid slottle
ifconfig eth1 192.168.1.99 netmask 255.255.255.0
```

B.2.2 Set up IPTABLES

You will need to add some rules to your firewall (iptables) script. Assuming the wireless interface is eth1, proceed as follows:

Load the required modules

```
modprobe iptable_filter
modprobe ip_queue
```

Configure iptables to queue all outgoing packets:

```
iptables -A OUTPUT -p icmp -j QUEUE
iptables -A OUTPUT -p TCP -j QUEUE
iptables -A OUTPUT -p UDP -j QUEUE
iptables -A FORWARD -p ALL -o eth1 -j QUEUE
```

It is important that all outbound traffic (on the wireless interface) are given the QUEUE target. Failure to do so will bypass the application and lead to collisions/performance problems. You may however add other rules to block unwanted outbound traffic.

B.3 Compile Slottle

B.3.1 Summary

```
./configure
./make
./make install
```

B.3.2 Requirements

To compile **slottle** you will need iptables installed. You will also need the development IPQ library (part of iptables) installed.

To do this, download the correct iptables source tarball (http://www.iptables.org/) to match your installed iptables (if you didn't install from a tarball). Do the usual "make" ("make install" if starting from scratch) and then "make install-devel".

B.3.3 Installation

Once you have iptables and IPQ installed you can compile **slottle** using the standard "./configure", "make" and "make install" commands.

There is a 'feature' in iptables which may cause make to fail. If you get errors like:

```
/usr/include/net/if.h:45: parse error before `0x1'
/usr/include/net/if.h:111: redefinition of `struct ifmap' etc
```

Then you need to edit /usr/include/linux/netfilter_ipv4/ip_queue.h and change "#include <net/if.h>" to "#include <linux/if.h>".

B.4 Starting the Application

B.4.1 Introduction

This section describes the command line parameters for **slottle**, a prototype application for Layer-3 Reservation TDM.

B.4.2 Quick Start Guide

If you wish to test the application as quickly as possible, without having to read much further, start here.

Configure IPTABLES (Section B.2.2) and ad-hoc mode (section B.2.1.)

slottle assumes reasonable defaults for most parameters. You will have to specify as follows:

```
sudo ./slottle [-i Interface] [-b Broadcast][-m] Nodenumber
```

NodeNumber A unique node number for each node

-i The name of your network interface (default eth1)

-b The Broadcast Address of your network

-m The first node to start on the network should use this parameter, to initialise the beacon. Subsequent nodes should not use it.

B.4.3 Synopsis

```
./slottle [-s] [-m] [-c SlotCount] [-d SlotDuration] \
[-l LogLevel] [-i Interface] [-p Port] \
[-b BroadcastAddr] Nodenumber
```

B.4.4 Description

slottle is a prototype application which implements Layer-3 Reservation TDM. It is designed for use in 802.11ad-hoc networks. **slottle** is able to alleviate hidden node problems, and to provide fair bandwidth sharing among nodes.

These preconditions must be met in order for the software to function correctly:

- **slottle** requires super user privileges. You can use the **su** or **sudo** to acquire these.
- **slottle** requires that IPTABLES be installed and correctly configured. The SETUPIPTABLES.SH script is provided for this purpose. See section B.2.2.

- It is advisable to configure your NIC for ad-hoc networking. Edit the SETUPADHOC.SH script, substituting the name of your NIC for eth1 where appropriate, and use it for this purpose. See section B.2.1.
- NodeNumber must be specified as a unique ID within the network. It is internally represented as a 32-bit signed integer.

The network interface name (-i) and broadcast address (-b) normally need to be specified. The defaults are **eth1** and **192.168.1.255**. Configure other options as appropriate.

B.4.5 Options

-m Set Master Mode. Because each slottle node tries to synchronise to a network before it transmits, some node in the network needs to transmit an initial beacon. A node started in Master Mode starts in status GOT_SLOT, which means that it will transmit beacons to which the rest of the network can respond.

-c N Set the Slot Count equal to N. This setting must be identically configured throughout the network.

-d N Set the Slot Duration to N. N is specified in microseconds (μs).

-l N Set the Logging level to N (refer to source for details)

-i Interface Specifies the wireless interface name. eg eth1,wl0

-p Port Specifies the port number

-b BroadcastAddr Specify the broadcast address of your network. Eg, 192.168.0.255.

B.5 The HTML Monitor

The **`slottle`** application attempts to write a status file, and updates it frequently. The file is in HTML format and may be viewed with a web browser. The file is stored in the following location:

```
/var/www/slottlestats.html
```

If a web server (such as Apache) is installed, the node can be remotely monitored via this file using a browser.

The output of the file shows information like this:

```
Node ID 999
Status GotSlot
Stats updated 14/12/2006 16:23:07
FrameID (age)   39172 (approx 0 hours, 6 min)
My latency  1802 µs, by method 20
Synced to Node  111
Sync Latency 2534 µs, by method 20
Successful Reservations 1
Errors: Slot Conflict  0
Errors: Rejected Reservations   0
Errors: No Slot Available      0
Errors: Beacon Received during Wrong Slot       0
Errors: Slot Lost 1
Error Rate      0.026 percent, (1 total)
Slot Table:   999 [_] [_] [_] [_] [_] [_] [_] [_] 111
Queue Packets sent, MBytes sent, Rate kB/s
High   0, 0.000, 0.000
Medium  782 , 0.086, 0.530
Low     0, 0.000 0,0.000
```

Appendix C

Accompanying CD-ROM

The accompanying CD ROM contains the following:

`Duff_MSc.pdf` is an electronic copy of this document.

`/References` contains electronic copies of references cited in this document, where available.

`/Source Code` contains the full C source code for the prototype application.

`/3rdParty` contains the Frottle and WiCCP source codes.

`/Experiments and Data` contains a variety of data captures and script files which were generated during the experiments.

References

[1] Ad-hoc frottle development page. http://www.melbourne.wireless.org.au/wiki/ ?MWR-PAdhocFrottleDev. [Accessed 20/12/2006].

[2] Community Owned Information Network, South Africa. http://csircoin.blogspot.com. [Accessed 17/03/2006].

[3] Durban Wireless Community Home Page. http://www.dwc.za.net. [Accessed 17/03/2006].

[4] IETF Differentiated Services (diffserv) WG Home Page. http://www.ietf.org/html.charters/OLD/diffserv-charter.html. [Accessed 20/12/2006].

[5] IETF Integrated Services (intserv) WG Home Page. http://www.ietf.org/html.charters/OLD/intserv-charter.html. [Accessed 20/12/2006].

[6] IPTABLES. http://www.netfilter.org. [Accessed 17/06/2006].

[7] Johannesburg Area Wireless User Group. http://www.jawug.za.net. [Accessed 17/03/2006].

[8] Locustworld. http://live.locustworld.com/index.php. [Accessed 24/08/2005].

[9] MIT Roofnet. http://pdos.csail.mit.edu/roofnet/doku.php. [Accessed 24/08/2005].

[10] Montreal's wireless mesh network. http://old.ilesansfil.org/wiki/Projects /MontrealWirelessMeshNetwork. [Accessed 24/08/2005].

[11] NTP (Network Time Protocol). http://www.ntp.org. [Accessed 20/12/2006].

[12] Pretoria Wireless Project. http://www.pwp.za.net. [Accessed 17/03/2006].

[13] Seattle Wireless. http://www.seattlewireless.net/. [Accessed 24/08/2005].

[14] The Champaign-Urbana Wireless Network. http://www.cuwireless.net/index.html. [Accessed 24/08/2005].

[15] The Frottle Home Page. http://frottle.sourceforge.net. [Accessed 17/12/2006].

[16] The IEEE 802.11 WG Home Page. http://www.ieee802.org/11/. [Accessed 21/12/2006].

[17] The LinkSys Home Page. http://www.linksys.com/. [Accessed 21/12/2006].

[18] The OpenWRT Wiki. http://wiki.openwrt.org/. [Accessed 18/11/2006].

[19] The Roofnet Tent City. http://pdos.csail.mit.edu/roofnet/doku.php?id=tentcity. [Accessed 24/08/2005].

[20] The UMA Technology Home Page. http://www.umatechnology.org/. [Accessed 20/12/2006].

[21] The Wi-Fi Alliance Home Page. http://www.wi-fi.org/. [Accessed 19/12/2006].

[22] WiCCP Home Page. http://patraswireless.net/software.html. [Accessed 20/12/2006].

[23] LinkSys WRT54G Wireless-G Broadband Router User Guide, 2005.

[24] ABRAHAMSON, N. The Aloha System - Another alternative for Computer Communications. In *AFIPS Conference Proceedings, Vol 36, 1970* (1970), pp. 295–298.

[25] ABRAMSON, N., AND KUO, F. *The ALOHA system*. Prentice-Hall, 1973.

[26] BHARGHAVAN, V., DEMERS, A., SHENKER, S., AND ZHANG, L. MACAW: a media access protocol for wireless LAN's. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications* (New York, NY, USA, 1994), ACM Press, pp. 212–225.

[27] BORGONOVO, F., CAPONE, A., CESANA, M., AND FRATTA, L. RR-ALOHA, a reliable R-ALOHA broadcast channel for ad-hoc inter-vehicle communication networks. In *Med-Hoc-Net 2002, Baia Chia, Italy.* (2002).

[28] BORGONOVO, F., CAPONE, A., CESANA, M., AND FRATTA, L. ADHOC MAC: new MAC architecture for ad hoc networks providing efficient and reliable point-to-point and broadcast services. *Wireless Networks 10*, 4 (2004), 359–366.

[29] CHATZIMISIOS, P., BOUCOUVALAS, A. C., AND VITSAS, V. Effectiveness of RTS/CTS handshake in IEEE 802.11a Wireless LANs. *Electronics Letters 40*, 14 (2004), 915–916.

[30] CLAUSEN, T., AND JACQUET, P. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), Oct. 2003.

[31] CONTA, A., DEERING, S., AND GUPTA, M. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443 (Draft Standard), Mar. 2006.

[32] DUFF, K., CLAYTON, P., AND TERZOLI, A. Synchronisation for IP-Layer TDM on Ad-Hoc Wireless Networks. In *Proceeds of South African Telecommunications Networks and Applications Conference (SATNAC) 2006* (2006).

[33] ELSON, J., GIROD, L., AND ESTRIN, D. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev. 36*, SI (2002), 147–163.

[34] GAMBIROZA, V., SADEGHI, B., AND KNIGHTLY, E. W. End-to-end performance and fairness in multihop wireless backhaul networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking* (New York, NY, USA, 2004), ACM Press, pp. 287–301.

[35] GEIER, J. 802.11 MAC Layer Defined. http://www.80211-planet.com/tutorials/articles, 2002. [Accessed 23/12/2006].

[36] GOFF, R. WiCTP: A Token-based Access Control Mechanism for Wireless Networks. Honours Thesis, School of Computer Science and Software Engineering, University of Western Australia, 2004.

[37] GOFF, R., AND A.DATTA. WiCTP: A Token-based Access Control Mechanism for Wireless Networks. In *Proc. International Conference on Computational Science and its Applications (ICCSA 2005)* (2005), pp. 87–96.

[38] HAAS, Z., AND DENG, J. Dual busy tone multiple access (DBTMA)- a multiple access control scheme for ad hoc networks. In *IEEE Trans. Commun., vol. 50, pp. 975–85, June 2002.* (2002).

[39] HEUSSE, M., ROUSSEAU, F., BERGER-SABBATEL, G., AND DUDA, A. Performance Anomaly of 802.11b. In *Proceedings of IEEE INFOCOM 2003* (San Francisco, USA, March-April 2003).

[40] LIN, X.-H., KWOK, Y.-K., AND LAU, V. K. N. A New Power Control Approach for IEEE 802.11 Ad Hoc Networks. In *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications* (2003).

[41] MILLS, D. Network Time Protocol (NTP). RFC 958, Sept. 1985. Obsoleted by RFCs 1059, 1119, 1305.

[42] MILLS, D. Simple Network Time Protocol (SNTP). RFC 1769 (Informational), Mar. 1995. Obsoleted by RFCs 2030, 4330.

[43] MILLS, D. Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. RFC 2030, 1996.

[44] NUTT, G. J. *Operating Systems: A Modern Perspective, Lab Update*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[45] OPENGROUP. The Single UNIX Specification Version 3, IEEE Std 1003.1-2001. WWW, 2001. http://www.unix-systems.org/single_unix_specification/ [Accessed 21/12/2006].

[46] PENTON, J. An Empirical, in-depth Investigation into Service Creation in H.323 Version 4 Networks. Master's thesis, Department of Computer Science, Rhodes University, 2002.

[47] PERKINS, C., BELDING-ROYER, E., AND DAS, S. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[48] PERKINS, C. E. *Ad hoc networking: an introduction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[49] POSTEL, J. Internet Protocol. RFC 791 (Standard), Sept. 1981. Updated by RFC 1349.

[50] RAO, A., AND STOICA, I. An overlay MAC layer for 802.11 networks. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services* (2005), pp. 135–148.

[51] ROBERTS, L. G. ALOHA packet system with and without slots and capture. *SIGCOMM Comput. Commun. Rev. 5*, 2 (1975), 28–42.

[52] S. KHURANA, A. K., AND JAYASYMANA, A. Effect of hidden terminals on the performance of IEEE 802.11 MAC protocol. In *Proc. 23rd Annual Conference on Local Computer Networks* (1998), pp. 12–20.

[53] SCHULER, D. Community networks: building a new participatory medium. *Commun. ACM 37*, 1 (1994), 38–51.

[54] SHENKER, S., PARTRIDGE, C., AND GUERIN, R. Specification of Guaranteed Quality of Service. RFC 2212 (Proposed Standard), Sept. 1997.

[55] WU, H., UTGIKAR, A., AND TZENG, N.-F. SYN-MAC: a distributed medium access control protocol for synchronized wireless networks. *Mobile Network Applications 10*, 5 (2005), 627–637.

[56] XU, S., AND SAADAWI, T. Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks? *Communications Magazine, IEEE 39*, 6 (2001), 130–137.

[57] YE, W., HEIDEMANN, J., AND ESTRIN, D. An energy-efficient MAC protocol for wireless sensor networks. In *Proc. IEEE INFOCOM* (2002).

[58] ZHU, C., AND CORSON, M. A five phase reservation protocol (FPRP) for mobile ad hoc networks. In *Proc. IEEE INFOCOM* (1998).