A Real Time Implementation of TBMAC using IEEE 802.11b

Mark Gleeson

A dissertation submitted to the University of Dublin, Trinity College

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

September 2004

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

Mark Gleeson

Dated: September 13, 2004

Permission to Lend and/or Copy

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

Mark Gleeson

Dated: September 13, 2004

Acknowledgements

I would like to thank my supervisor Prof Vinny Cahill for his advice and guidance throughout the dissertation.

A special thanks to Barbara Hughes who made TBMAC in real-time possible, and for all her help and guidance through the year. Thanks as well to all those in the distributed systems group who helped along the way with there time, advice and the use of there laptops.

Finally I would like to thank the Networks and Distributed Systems class of 2004 who made the experience a thoroughly enjoyable one.

Mark Gleeson

University of Dublin, Trinity College

September 2004

Abstract

Emergency services, and automated vehicle platoons are frequently used examples of the need for real-time ad hoc communication, which demand that communication be time bounded overall.

The medium access control protocol regulates access to the medium. Time division protocols give deterministic, time bounded delivery of messages. TDMA, time division multiple access forms the basis of the time bounded medium access control protocol, TBMAC. TBMAC provides every mobile station with predictable access to the medium with known probability and bounded delay.

Currently TBMAC has been simulated using the NS2 simulator. This dissertation implements TBMAC using RTAI real-time Linux and IEEE 802.11b as a basis. The implementation supports a non empty cell with a static cell membership. The implementation makes use of the broadcast tightness property to achieve tight clock synchronisation between multiple stations. The TBMAC protocol was implemented in real-time using the TDMA MAC layer during the dissertation.

Simulations have been relied upon in the evaluation of new MAC protocols. This dissertation evaluates the TBMAC protocol in the real world and compares its performance to that of the CSMA MAC protocol. Real world experiments give a much better and more realistic view of the performance of the MAC protocol compared to experiments carried out in a simulator.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Wireless networks have come of age in the last decade, widespread adoption of standards such as GSM[33] and IEEE 802.11[22] has led to rapid wide-scale deployment. The sheer nature of wireless networks has lead to a complete rethink about how networks are structured and work.

Traditionally networks where infrastructure based, controlled typically from one centralised point, wireless network introduced the concept of infrastructure less networking, ad-hoc networking. This has required extensive research in the field of ad-hoc routing protocols. Wireless networks are significantly different in nature and present significant challenges compared to wired networks[21]. While many assumptions of wired networks do not apply in the ad hoc domain, the basic CSMA medium access control protocol has remained in use with little alteration. CSMA is not suitable for urgent time sensitive traffic[8]. While much work has tried to overlay quality of service facilities upon 802.11[35, 26], the underlying lack of determinism in the medium access control (MAC) protocol has not been addressed.

This dissertation aims to implement the TBMAC protocol [15, 13]. TBMAC aims to provide a time bounded medium access control service for IEEE 802.11 based ad-hoc networks. This service provides time bounded access, with well known probability to the wireless

medium. Long discussed applications communication for emergency services at major accident sites[29] and automated vehicle platoons[14] are now realisable using TBMAC. Starting with a time bounded medium access service paves the way for further services such as quality of service facilities and it further opens the possibility of the use of IEEE 802.11 in safety critical systems, where deterministic and timely message delivery are essential such as in a automated/driverless vehicle[14].

1.2 What is Medium Access Control ?

When computers are connected together to form a network, they share access to the network. At the physical layer the underlying network shares a transmission medium on which the bits that make up that data being exchanged travel. There is single communication channel that is shared by all[6]. Various technologies have been developed the most common of which rely on the use of optical (FDDI, gigabit Ethernet) electrical (Ethernet, Token Ring) and wireless (GSM, HIPERLAN, 802.11, Infra-red) transmission mediums. They all share one common characteristic, in general only one station can send information at a time. Such networks are known as broadcast networks. Many stations can potentially see packets transmitted to stations other than themselves, 802.11 and first generation Ethernet are both examples of broadcast networks.

To support many stations wishing to transmit concurrently, some kind of coordinating function is required. This co-ordination function is known as the medium access protocol or MAC. The MAC protocol is the protocol used for access to the medium (the radio link in a wireless network) with the resulting transmission of data on that medium[27]. It defines how and when stations may access the medium[8].

A key requirement is that to provide access to the medium within a bounded time known a priori, this is of upmost importance. In the face of a finite resource (channel bandwidth) an admissions control policy is also an essential feature together with bandwidth regulation to ensure every station gets the bandwidth it requires as far as is practicable. A key issue

is contention, contention for the medium wastes bandwidth. A time bounded deterministic approach to contention is required. A collision is outcome of poor and inefficient use of the medium, if collisions are eliminated using a TDMA or FDMA[1] approach, represents a more efficient use of the wireless medium.

1.3 Proposal

TBMAC is a recently developed time bounded medium access control protocol[13]. The goal of TBMAC is to provide predictable deterministic access to the wireless medium by providing *dynamic but predictable slot allocation*[15]. Access to the medium is predicable and is time bounded with a well know probability[13]. TBMAC has been subject to extensive simulation in the NS2[5] simulator to verify these theoretical bounds are achievable.

Unlike much previous work in the area of wireless MAC protocols this dissertation will rely on results gathered from a real world implementation not from simulations as the vast majority of work in this area has done so in the past[7, 12, 26, 29, 35]. While simulations are a powerful tool they do not represent the actual performance of a MAC protocol which may be effected adversely by interference and other effects.

The primary aim of this dissertation is to take the design of TBMAC and 802.11b and to deploy and evaluate its performance under real world conditions using RTAI[3] real-time linux as an experimental platform. In order to implement TBMAC tight clock synchronisation will also be implemented. This dissertation will provide a working implementation of TBMAC. This implementation will be evaluated and the challenges and practicalities of a real world implemented discussed.

1.4 Scope

The scope of this dissertation is limited to the implementation of the TBMAC protocol on the RTAI real-time linux platform. All evaluation of TBMAC and comparative analysis will

[1] Frequency division multiple access, each station is allocated its own frequency

be conducted using 802.11b based devices and will use RTAI.

1.5 Aims

The aims of the dissertation are to,

- To implement and evaluate a clock synchronisation protocol of high accuracy.
- To implement static TDMA and to evaluate its performance in the real-world.
- To implement the TBMAC protocol using RTAI real-time Linux.
- To evaluate the performance of TBMAC in comparison to that of IEEE 802.11b under real world conditions.

1.6 Dissertation Roadmap

The rest of this dissertation report is laid out as follows,

Chapter 2 State of the art, provides a review and discussion of existing wireless medium access control protocols. The issue of clock synchronisation is also discussed owing to the reliance of TDMA style MAC layers on tight clock synchronisation.

Chapter 3 Background, A brief overview of the TBMAC protocol and the rtcomm wireless card driver.

Chapter 4 Design, Gives an overview of the design and structure of the proposed system.

Chapter 5 Implementation, this chapter will discuss the final design and implementation of the components that together implement and support the TBMAC protocol.

Chapter 6 Evaluation, this chapter will discuss the experiments carried out and results.

Chapter 7 Conclusions and future work, this chapter summaries the work presented and points to possible further extensions and improvements.

Chapter 2

State of the Art

In this chapter I will discuss a number of approaches to the medium access control problem. Based on this analysis it is clear that a time division multiple access (TDMA) style MAC layer is required to satisfy both the time bounded and deterministic scheduling required for real-time traffic. The second half of the chapter will discuss a prerequisite required to develop and test an ad hoc TDMA MAC protocol, namely tight and high resolution clock synchronisation.

2.1 Co-ordination & Distributed Agreement

A major challenge in the development of a efficient MAC protocol for use in wireless networks is achieving distributed agreement, confronted with the lack of global knowledge and the high probability of failure of both stations and individual messages. Ad hoc networks are highly dynamic. Communication between stations cannot be assumed to be symmetric[12] nor are all stations fully connected, ad hoc networks operate on a peer to peer basis[21]. This highlights some significant challenges which are not present in the wired domain.

2.1.1 What is a collision

A collision occurs when two (or more) packets are transmitted at the same time, this results in the loss of the packets involved. The understanding and perception of a collision is key in the development of a successful MAC protocol. Bharghavan et al in[7] provides an insight into collisions in the wireless domain. Unlike in the wired domain collisions occur at the receiver not the transmitter, this is a very important distinction. Neither the transmitter or receiver actually can determine if a collision has occurred thence an explicit response packet is desirable. A collision can be used as the generic explanation for all failures on a wireless network. Since a station is unable to detect a collision on it's own, a collision is in fact a omissions failure. However the loss of a packet could mean many things,

- Could be a legitimate collision
- A station could have gone out of range
- Noise on the channel may have corrupted the signal
- A station may have failed for some reason most likely a weak battery
- The station may decide for other reasons not to reply, this can be used as a form of admission control as used in [29]

The response to each of these failures is the same even though the causes are very much different, for instance if the packet was lost due to excessive noise on the channel the MAC protocol would assume a collision and the back off counter would be incremented even though there isn't any contention on the medium.

Hidden Terminals

A major issue in wireless networks is that of the detection of collisions. Wireless networks are location dependent, while the medium may be idle at station A it may be active at station B. In fig. 2.1 station B is within the transmission range of stations A and C but A cannot hear C nor can C hear A. So stations A & C may see the medium clear and may transmit, a collision occurs at B. A and C will time-out waiting for an acknowledgement and resend the

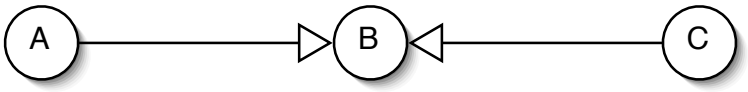

Fig. 2.1: Classic hidden terminal problem

packet. In this situation C is the hidden terminal relative to A and vice versa.

In a effort to avoid this problem some kind of co-ordination function is required, collision avoidance. Typically for large packet sizes the re-transmission overhead is significant and the overhead of the a small initial exchange can be justified. The solution is to ensure that all stations in range of both the transmitter and receiver are aware of the transmission before it occurs. A hand shake is used. The transmitter sends a 'request to send' (RTS) packet to its receiver which then replies with a 'clear to send' (CTS) packet, these packets contain the expected size of the imminent packet transmission. All stations in range of the RTS-CTS exchange, which are the only stations that can collide with the transmission defer allowing the data packet and acknowledgement to proceed with a very much reduced possibility of collision. The overhead of sending a RTS-CTS is minimised when packet size is large. [12] presents the results of a simulation using 802.11 technology, the results indicates that a threshold of between 250 and 400 bytes should be used before the RTS-CTS handshake is used. If the initiating station does not receive a CTS response it assumes a collision has occurred, backs off and re-contends. This introduces a possible major inefficiency, what happens if the CTS is sent and not received, all stations in range of the RTS-CTS exchange have backed off assuming that the medium would be busy during that period.

2.2 Approaches to the MAC Problem

Current MAC approaches can be broken into a number distinct classes,

- Ad-hoc. Contention-based structures, for every packet the station contends[7, 22].

- Contention free. These are slotted techniques, each station has a specific time or frequency slot(s) in which it alone can transmit[1, 33].
- Contention free with a contention period. A combination of the two above techniques, allowing flexibility while retaining the deterministic scheduling found in slotted protocols[13, 22].

2.2.1 Ad hoc, Contention-based structures

This class of MAC protocol are based heavily on maximising fairness of access, every station has an equal chance of successful transmission. Each station must contend for every single packet they wish to send. Such an approach is acceptable given the potentially highly unstable structure of a wireless network, it avoids the need to share or store information and requires no explicit agreement between stations. Two distinct classes of MAC protocols exist, MACA and CSMA. While they both deal with medium access in different ways they both deal with contention and collisions in a similar fashion using a exponentially increasing back-off counter.

CSMA, Carrier Sense Medium Access

In this class of MAC protocol the medium is sensed for a small amount of time, the slot time. This is known as physical medium sensing. If there are no other stations heard to be transmitting over one full slot time, the medium is deemed to be idle and the packet may be sent. A positive acknowledgement scheme is required, since it is possible for two stations to view the medium idle and begin to transmit at the same time. The slot time is longer than that of the gap between a data packet and acknowledgement packet so that ongoing transmissions are not disrupted. Other stations intending to transmit will not see the medium idle for a sufficient length of time to permit transmission. In its basic form CSMA suffers from the hidden terminal problem.

MACA, Medium Access Collision Avoidance

Karn[28] proposes the MACA protocol in an effort to avoid the hidden terminal problem that effects CSMA. MACA relies on prior knowledge of other traffic to determine if a station can transmit. When a station wishes to transmit it first checks to see if it knows of any stations transmitting at this time, this is referred to as virtual carrier sensing. First a 'request to send' (RTS) packet is sent, this packet contains the length of time the medium will be in use. All stations in range of the RTS packet freeze their back-off counters for this period contained within the RTS packet. The destination responds with a 'clear to send' (CTS) packet. The CTS serves a dual purpose. It indicates to the transmitter that the RTS was received and indicates the length of the data packet. All stations in range of the receiver may back-off to avoid a collision. This dialog prevents a hidden terminal problem as discussed in section 2.1.1. Since the RTS-CTS exchange is used it is assumed that there will not be a collision as all stations in range of the RTS-CTS exchange know the medium is in use, so no acknowledgement is sent since it is deemed unnecessary.

MACAW

MACAW[7] is an extension of the MACA protocol. Bharghavan et al proposes number of extensions to improve the performance of the MACA protocol in terms of throughput and fairness. Standard MACA does not use an acknowledgement packets relying entirely on the RTS-CTS handshake to protect the data packet. MACAW adds an acknowledgement packet, while this reduces the maximum throughput, it provides a much superior throughput under poor channel conditions. MACAW proposes a solution to the exposed terminal problem, since collisions occur only at the receiver an additional packet is sent between the receipt of the CTS packet and the transmission of data packet any station in range of this data-sending (DS) packet know the RTS-CTS was successful and defer until the acknowledgement (ACK) packet passes.

The use of the RTS-CTS-DS-DATA-ACK handshake, once again leads to a minor reduction in throughput but leads to a far fairer distribution of bandwidth among contending

Fig. 2.2: The exposed terminal problem

stations.

CSMA/CA, Carrier Sense Medium Access with Collision Avoidance

CSMA/CA combines the MACA approach with that of CSMA. Both CSMA style physical and MACA style virtual medium sensing are used. If the medium is known to be busy it is not sensed otherwise, CSMA is used to check if the medium is in fact not in use. Optionally then a MACA style RTS-CTS exchange may take place. Collisions are minimised, though given the random nature of medium access it is still possible for two stations to transmit a RTS packet at the same time. Unlike MACA acknowledgement packets are used. This is MAC protocol used in the popular IEEE 802.11 standard[22].

2.2.2 Contention-free, slotted techniques

These are high performance contention free MAC protocols. This class of MAC protocol are typified by there master slave behavour[1, 33]. There is typically a controlling station known as the base station which manages bandwidth allocation as well as performing admission control. Time is divided into a discreet number of pieces known as slots, all slots are of equal length. Only one station may transmit in a slot at a time. Such a MAC protocol is known as time division multiple access, TDMA. Allocation of slots is typically handled out of band, a dedicated communication channel is used over which slots are requested.

In all slotted structures there is a need for precise co-ordination between stations to ensure the slots align correctly a small space know as the guard space is left between slots to allow for a small misalignments in synchronisation and also to allow for propagation delays caused by stations being a variable distance apart. Synchronisation in TDMA is one of logical time,

all that is required is a reference to the start of the next cycle. A slotted structure showing the guard space is shown in fig. 2.3.

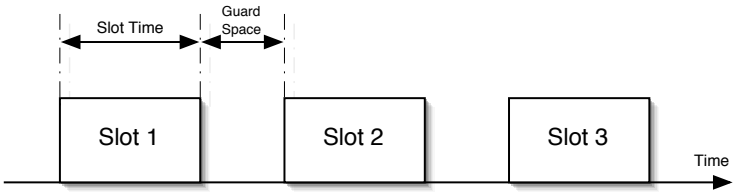

Fig. 2.3: Interslot gaps in TDMA

A heartbeat signal provided out of band is typically used to gain the necessary synchronisation. For example in the GSM mobile system the base station broadcasts a clocking signal on a dedicated frequency channel[33], known as the synchronisation channel (SCH) which all devices in it's coverage can receive and use to synchronise. While these networks are dynamic in the sense that station may join and leave at any time they do not support ad hoc, peer to peer behavour. Failure of the base station or loss of contact with the base station results in disconnection and failure of the client station.

2.2.3 Contention-free with a contention period

In common with slotted techniques these are connection oriented protocols designed to offer the performance of a fixed slotted network but still allowing significant flexibility in dynamic allocation of slots. Dynamic techniques attempt to merge the flexibility of the contention based approach with the performance of true TDMA. All transmission occurs on the same channel.

A number of papers have proposed various approaches [29, 34, 35] to the need to support real-time traffic. Some are application dependant and rely on feature of a specific traffic type, such as MPEG video's ability to tolerate lost packets[37]. Gerla et al[29], proposes a multi-hop network providing support for real-time traffic. The approach retains a MACA MAC protocol and overlays a windowed structure, an initial RTS-CTS exchange is used to

allocate a transmission window, if no slots are available the receiving station ignores the RTS and does not reply with a CTS, such an approach provides effective admission control. In addition slots are deallocated if they go unused.

While the use of an RTS-CTS exchange minimises the possible impact of the hidden terminal problem, the RTS-CTS exchange is only used to set up the reservation, under normal conditions only a DATA-ACK dialog takes place. In addition each station only allocates windows it hears about in the RTS-CTS exchange so if a station moves all of a sudden it may find that its windows now conflict with others, [29] further proposes that the reservation table be propagated periodically, however how to deal with conflicting slot allocations is not discussed, this concern is particularly important if there are frequent updates in the topology of the network.

Many proposals to support real-time traffic rely on the stations on the network finding a stable state [8, 26, 34]. Following an initial disturbance the system will, subject to a number of initial conditions related to the minimum packet size and the number of stations on the network, since there is no admissions control no upper limit is placed on the number of stations so the time bound is proportional to the number of stations which may not be known in advance nor may it even be known are runtime. With a TDMA based structure the maximum number of stations is known and fixed and forms a significant design decision when the system is built.

In a TDMA environment the maximum number of slots that can be allocated is related to the expected number of stations, the data rate each station will generate and the speed of the transmission medium. A major issue is that the bulk of research in the area is based on adapting the MAC protocol (CSMA) to cope with continuous bit rate traffic such as voice and or video. Another approach is to use the standard MAC protocol and reserve a slot some time in the future[35, 29]. Both still rely on the underlying MAC protocol which remains non deterministic in the face of contention or collisions.

2.3 An Overview of IEEE 802.11

The 802.11 standard comprises two parts. A number of physical layer specifications, 11a, 11b, 11g and a single CSMA/CA MAC protocol. 802.11 was designed as a direct replacement for ethernet and is sometimes referred to as wireless ethernet. This section will discuss both 802.11 layers.

2.3.1 Physical Layer

The current set of 802.11 standards[22] define both a radio physical layer as well as infra red. There are currently four approved 802.11 physical layers. The initial 802.11 standard (1997), commonly known as wavelan described a 1 Mbps, optionally 2 Mbps wireless physical layer, this standard evolved in 1999 into 802.11a[23] and 802.11b[24]. 802.11a runs in the unlicensed 5 Ghz band using a frequency hopping spread spectrum transmission technology to achieve a potential throughput of 54 Mbps, however due to the higher frequency used transmission range is shorter than that of the other standards. 802.11b which runs in the 2.4 Ghz ISM[1] band using a direct-sequence spread spectrum transmission technology. Its throughput is limited to a potential maximum of 11 Mbps. 802.11b was the most successful standard due to its ease of deployment plus its much superior range over that of 802.11a. Notably 802.11b retained backwards compatibility with the earlier 802.11 (1997) wavelan standard. 802.11g evolved from 802.11b but uses a more efficient coding scheme to obtain a potential throughput of 54 Mbps thus matching that of 802.11a while retaining compatibility with 802.11b.

This dissertation will concentrate on the more popular 802.11b standard. While the MAC layer is separate from the physical layer, the design of the MAC protocol may need to acknowledge certain traits of the underlying physical layer, in particular the supported data rate(s) and inter-frame spaces. It should be noted that regardless of the data rate used in 802.11b (& 802.11g) a portion of the packet header is still sent at the base rate of 1 Mbps to

[1]ISM, Industrial Scientific and Medical band, a license free piece of radio spectrum

ensure compatibility with all stations regardless of which transmission rate(s) they support. A station may choose to use a lower rate when channel conditions are poor.

2.3.2 Medium Access Control Layer

Regardless of the physical layer used the same MAC protocol is used. The 802.11 standard defines two MAC protocols, one for use in ad-hoc networks and one for use in infrastructure networks. The infrastructure MAC protocol is built upon the ad-hoc MAC protocol. The 802.11 standard defines three inter-frame spaces, it should be noted that regardless of the transmission speed these inter frame gaps remain the same.

- The Short IFS or SIFS, 10μsec
- The Point co-ordination IFS or PIFS for short, 30μsec
- The Distributed co-ordination IFS or DIFS, 50μsec

These are used to prioritise media access, acknowledgement packets have the highest priority and use the shortest inter frame space, the SIFS interval. In ad-hoc mode the longest interval, the DIFS interval is used. A station must see the medium as idle for a DIFS interval before it may transmit.

2.3.3 Ad-hoc Mode

The 802.11 standard prescribes the distributed co-ordination function, or DCF for short. Two types of carrier sensing are employed, physical and virtual. Physical is employed when directly listening to the medium, the medium must be idle for a minimum of a DIFS interval before the station can transmit. Virtual carrier sensing is employed when the station has prior knowledge of an upcoming transmission. The RTS and CTS packets contain information about how long the next packet will be. All receiving stations update the network allocation vector, NAV and cease contending for the time indicated in the RTS and CTS packets. Stations only start to re-contend once the length of time indicated in the NAV has elapsed.

2.3.4 Infrastructure Mode

The base station maintains a list of stations which are associated with it, it periodically polls each of the stations in turn. This polling period is further broken down in to a contention free period (CFP) and a contention period (CP). The CFP starts with the access point sending a beacon frame. The length of the CFP is flexible and is decided upon by the access point depending on the current level of traffic it sees. The length of the CFP is carried in each beacon frame. Stations update there NAV to the length of the CFP. During the CFP a station may only transmit if it has been polled by the access point.

The access point polls each station in turn, while in theory this gives a slotted structure, the access point is in control and regulates access. Such a protocol has a single point of failure and is of no use in an ad hoc environment, however the CFP and CP idea is used in TDMA style systems such as [29] to allocate slots without effecting existing traffic flows.

2.4 Adapting 802.11 through the inter-frame spacing to support real-time traffic

[26] Sobrino et al proposed an interesting approach, which retains compatibility with current 802.11 devices. A two stage approach is proposed, firstly how to prioritise access and secondly how to introduce determinism in resolving collisions.

The 802.11 standard as discussed earlier, defines three inter-frame spaces, of which only two (SIFS & DIFS) are used when running in ad hoc mode. Sobrino proposes that the unused inter-frame space (PIFS) could be used to prioritise traffic. If urgent traffic used a shorter interface spacing, that traffic could gain priority access to the medium. Such a solution allows two distinct traffic types real-time and non real-time. Real-time traffic contends for the medium using the shorter inter frame space and in doing so prevents any non real-time traffic from gaining the medium since it (the non real-time traffic) must observe the medium to be idle for longer than the real-time stations.

Contention is still non deterministic, so in addition a black-burst contention protocol is

used to achieve agreement. The contention protocol relies on stations transmitting a jamming signal. The length of this jamming signal known as a black burst is proportional to the length of time the station has been waiting to transmit. By sending a longer burst than any other station contending at the same time a station may acquire the medium.

While a promising solution a number of areas are not addressed, since the black burst is proportional in duration to the time waiting a significant amount of bandwidth may be wasted in the contention phase. Packet exchange is connection-less but the traffic is connection oriented. Each packet must explicitly contend for the medium each time. The outcome of the contention process is now deterministic, the delay before the packet is actually sent is not. An important question, is what happens if two real-time stations decide to transmit at the same time, since the protocol is deterministic the longest waiting station must win, but what if two or more stations have been waiting the same length of time, following there black bursts they will both see the medium as being free, assume they have won and attempt to transmit which will result in a collision.

In an effort to maximise data throughput no acknowledgements are sent, the use of negative acknowledgement scheme is not suited to the high packet loss often seen on wireless networks especially since the network is dynamic and more failure prone there wired equivalent[21]. If a packet fails to arrive a data packet is sent requesting re-transmission. Data packets can only be sent if there isn't any real-time traffic. In this scenario what would be assumed to be an urgent request to retransmit a possibly critical packet is given the lowest possible priority.

Stability is a key requirement in a real-time environment, [26] contains an exhaustive proof of stability. Stability is a function of several variables, packet size, number of stations and the expected data rate. There is an upper bound on the number of stations for a given level of traffic, for which the medium access will remain stable. There is no distributed co-ordination nor is there any admission control. It is conceivable that the upper bound on the number of stations could be breached quite easily.

2.5 Hyperlan2

Both Hyperlan2[1] and IEEE 802.11a share a very similar physical layer, but have very different MAC layers. Hyperlan was developed in part by ETSI, the European body for telecoms standards, there influence can clearly be seen in the design and functionality of the Hyperlan platform. The goal behind Hyperlan was to provide next generation (beyond 802.11) high speed local area wireless networking while retaining the structure of a cellular network.

Hyperlan is a connection oriented protocol, every connection can be assigned a specific level of quality of service *in terms of bandwidth, delay, jitter, bit error rate*[27]. This is a key difference from 802.11. To provide such guarantees Hyperlan uses a TDM based structure[27]. Unlike 802.11 which provides support for only ethernet. Hyperlan provides support for the 4 ATM[2] traffic classes namely constant, variable, available and unspecified bit rate. The convergence layer is key to power of Hyperlan, it is a generic architecture which allows services to be added. Support is provided for Ethernet, IP, ATM and UMTS[3] traffic. The convergence layer supports both cell and packet based communication.

Unfortunately Hyperlan was launched following 802.11 which already had significant deployment. Results published in [17] clearly show the benefits of its time division structure. Efficiency remains constant at all transmission speeds while a comparable 802.11a device quickly falls off as throughput increases.

2.6 Techniques from Sensor Networks

Carley et al in [8] describes a wireless network consisting of small low powered sensor devices. This network uses TDMA scheduler to schedule transmissions. While no comment is made on actual performance, the paper provides significant insight into the challenges of storing and maintaining the TDMA table. Up to 50% reduction in throughput over the actual bandwidth

[2]Asynchronous Transfer Mode, a virtual circuit switched technique used extensively in the telecoms industry

[3]Universal Mobile Telecommunications System, the latest evolution in mobile telephone systems, also known as 3G

is quoted. The main reason for this overhead is to do with clock synchronisation, the inability to achieve extremely tight synchronisation. In addition all TDMA style implementations require a guard space a small gap in time between transmissions to avoid any potential overlap caused by propagation effects and poor synchronisation. 50% is quoted as the upper bound, in a time bounded system it must be assumed to be true at all times and the system designed accordingly. Pure TDMA type approaches also suffer from this problem, however owing to the typically centralised controlling station synchronisation is somewhat easier to achieve. Experiments concentrate on the TDMA technique used.

However no discussion is made of how the approach copes with the dynamic addition or removal of sensors. A critical issue it that it must be possible to actually generate a schedule. While in a sensor based environment, data rates for each device is likely to be known in advance. Such a restriction is extremely important when considering a dynamic multi hop ad-hoc typology.

2.7 Overview of Features in a Selection of Wireless Technologies

This section will present a feature comparison of a selection of current wireless technologies which have been discussed in this chapter. They are all evaluated under 7 conditions,

Distributed Is the protocol distributed ?

Technology The underlying MAC protocol, MACA, CSMA or TDMA

Dynamic A key criteria does the protocol allow the dynamic addition and removal of stations

Efficiency What portion of the available bandwidth is available for actual useful data traffic

Hidden terminal Does the protocol adequately deal with the hidden terminal problem

Coordination Is contention for the medium coordinated

	Hyperlan2	802.11 DCF	802.11 PCF	MACAW	B Bursts	TBMAC	Sensors
Distributed	NO	YES	NO	YES	YES	YES	YES
Technology	TDM	CSMA/CA	CSMA/CA	MACA	CSMA/CA	TDMA	CSMA
Dynamic	YES	YES	YES	YES	YES	YES	NO
Efficiency	78%	?	?	?	?	?	50%
Capture	YES	NO	NO	NO	?	NO	NO
Hidden Terminal	NO	NO	NO	NO	NO	NO	NO
Co-ordination	YES	NO	YES	YES	NO	YES	YES
Deterministic	YES	NO	YES ?	NO	YES	YES	YES

Table 2.1: Comparison of a number of current wireless technologies

Deterministic Is media access deterministic

Table. 2.1 shows the results of this comparison. It is clear to note that support for deterministic MAC behavour in the ad hoc environment is poor.

2.8 Clock Synchronisation

Clock synchronisation consists of three separate requirements, how is the clock represented and stored on the device? How is agreement achieved to what the actual time is? How the time is altered on the device? Time is monotonically increasing, this statement encompasses two issues inherent in time, it can never go backwards nor can it stop, it may however slow down[11].

2.8.1 Support in Current PCs

Support for high resolution clocks in current off-the-shelf desktop Intel IA32 hardware[4] is poor and has seen no improvement since 1984. Prior to 1984 desktop PCs did not have hardware clocks and required the clock to be reset at every start-up. In 1984 a hardware/BIOS clock was added, the clock is driven by a Motorola 146818 or compatible chip which maintains a

[4]This refers to all computers using 32-bit processors from the Intel x86 family or compatible

32-bit counter, storing time to a resolution of 1sec since the epoch date of 1/1/1970. During the very early phase of initialisation the value of this clock is copied and is used as the starting point for the software clock.

Software Support

The software clock is fully independent from the hardware clock. On uniprocessor PC hardware it is supported by an Intel 8254 (or compatible) programmable interrupt timer (PIT), which issues a well known interrupt, typically every 10ms. The host OS uses this as the basis for its software clock. This approach is clearly subject to significant variation owing to variable delays in processing this interrupt.

Hardware Support

Pentium class, and other modern processors families, contain a special read-only 64-bit hardware register accessible through the `rdtsc` instruction[25]. The processor increments the contents of this register on every clock cycle, providing an extremely fine grained clock, e.g. for a processor speed of 1Ghz this provides 1ns resolution.

Recent versions of `/arch/i386/kernel/time.c` file, the code which implements the software clock retains the use of the interrupt timer (PIT) to maintain the software clock but uses the tsc instruction to correct for variances that occur while using the PIT, owing to both inaccuracies in the PIT and variable delays caused by interrupt processing.

2.9 Synchronisation Protocols

Many synchronisation protocols have been proposed, which typically involve devices exchanging messages, while in some way trying to determine the round trip time between stations. The round trip time provides an estimate of the time that will pass from a message leaving one station and receipt on the other. This is a critical requirement. More recent work has tried to exploit the tightness property of broadcast networks where a message sent from one

station is received at the exact same instant at all stations.

2.9.1 NTP, The Network Time Protocol

NTP, the network time protocol[2] by Mills et al is the most widely used clock synchronisation protocol on the Internet. Its hierarchical structure scales well and its ability to detect rogue clocks and deal with clock drift are welcome and useful features. However NTP is typically implemented as a user-level application, and thus messages it sends and receives are subject to higher delays than an implementation at kernel level. NTP uses a 64-bit value for the clock, this offers 233 pico second (10^{-12}sec) resolution. Primary level, level 1 devices are synchronised to a precise external clock, GPS or a caesium clock for example. The key to NTP's accuracy is its advanced adaptive hybrid phase-lock/frequency lock loop algorithm with nearly 20 years of evolution. While accuracy to the order of a few micro seconds is quoted [30]. To obtain these levels of accuracy requires that NTP be tightly integrated into the kernel with an external stable oscillator to provide a stable source of clocking information.

2.9.2 GPS, The Global Positioning System

In addition to providing location services, GPS also provides a synchronised clock GPS service. GPS suffers from one major restriction, GPS requires that the receiver be in the range of several satellites. Use in urban areas where buildings may obstruct the line of sight to the satellites may not be possible, GPS is unusable indoors. High end receivers offer accuracy better than 200nsec. While such receivers may be expensive, the key concern is that of the GPS device itself. In a sensor network for example adding such a device will increase the device size and power consumption both of which are key restricting factors in portable devices.

2.9.3 Physical Layer Broadcast

A number of recent papers [18, 32] had discussed the potential of the broadcast functionality found in many network technologies such as Ethernet and 802.11 in the development of clock

synchronisation protocols. The time to send a message on a CSMA or MACA network is not deterministic but its reception at the destination is subject to much less variability because the propagation delay is ignored. A message from one station will be received at all other stations[5] at what effectively will be the same instant. Stations may then exchange messages concerning the clock synchronisation packet was received. Based on this the clocks may synchronise. By removing the send process from the protocol a huge amount of possibly variable delay is removed from the critical path of the protocol. In addition the packet sent does not actually contain any clock information, all that is required is a sequence number and a flag to indicate that the packet is to be used in the synchronisation process. Such information can be easily piggybacked on normal traffic, and is small enough to fit inside the packet header.

Clearly to make effective use of a slotted or windowed MAC layer there needs to be tight co-ordination between all stations as to the exact starting time and length of each slot, achieving such tight co-ordination is particularly difficult in an ad-hoc network. Infrastructure based networks have the benefit of a single coordinating station to whom all stations can synchronise themselves, thus clock synchronisation is relative to that base station only. Often an out of band signal is used, such as the SCH used in GSM[33] where a specific frequency channel is allocated for this sole purpose. In this case clock synchronisation is not strictly required, all the stations need to know when the start of next cycle is. Recent work presented in [32] shows that tight clock synchronisation is possible in multi-hop ad-hoc network even under significant packet loss.

The removal of the send delay from the critical path brings a key advantage over NTP style systems, under heavy load CSMA and MACA type networks suffer from significant delays this variability adds greatly to error. While the traditional approach is transmitter centric, with physical layer broadcast the time of arrival at the receiver is key.

[5]It is important to note that a station cannot transmit and listen on the same channel at the same time

2.10 Conclusions

It is quite clear that current MAC protocols cannot cope with the demands of time critical data. A significant body of work has been done in the area of MAC protocols in the recent past, however there are two major areas where work is lacking. The vast majority of work published in this area relies in its entirely on simulations[7, 12, 26, 29, 35], typically using NS2[5] or similar. While simulators are a powerful tool, they work based on a simplified set of rules describing the environment in which they communication take place. This work while valuable does not address the root cause of the problem, the CSMA based medium access approach.

TDMA comes at a price, as outlined in [8] just storing the TDMA table can be a significant burden on resources. While in CSMA there is no hard limit on the number of stations, the number of slots available in a TDMA solution is fixed and known in advance. So while CSMA gives flexibility in both the number of stations and in the packet sizes they can use, performance drops off rapidly as the number of active stations increase, while in the case of TDMA the performance remains constant regardless of the offered load[17]. TDMA is deterministic and is recognised as being suitable for supporting real-time communication [31, ch 16]. Hyperlan was an extremely promising technology, its time division based structure, connection oriented nature and extensive support for quality of service parameters. It is let down by only one weakness its need for a dedicated (master) base station.

The emphasis in many papers[37] is on support for multimedia, many solutions leverage the fact that the traffic is tolerant of packet loss. Changing inter-frame spacing and contention windows may improve performance under specific conditions but do nothing to address the fundamental lack of determinism in the MAC layer. A lot of work is top down in it approach overlaying quality of service over a weak MAC layer instead of the bottom up approach, which starts by providing a time bounded MAC service upon which higher layers may implement time bounded services. In a situation where performance is the key driver this approach is to be favoured.

Chapter 3

Background

This chapter will provide the background to recent and current work that this dissertation will make use of during the implementation and evaluation stage. Two elements are discussed. TBMAC is the medium access protocol that will be implemented later in this dissertation. Real time access to a wireless card is essential in the implementation of TBMAC which will be provided by the rtcomm driver.

3.1 TBMAC, The Time Bounded Medium Access Control Protocol

TBMAC is a new medium access protocol for ad hoc wireless networks[13, 15]. It is generic in design and does not specify a underlying physical layer. TBMAC is based on the TDMA MAC protocol offering a contention free environment with deterministic packet transmission delays. TBMAC implements a TDMA like MAC protocol in an ad hoc wireless network structured in a cellular fashion. The TDMA MAC protocol is adapted by dividing the TDMA cycle into two pieces, the contention free period (CFP) and the contention period (CP). Stations determine there location through the use of GPS and set there wireless interface to listen on the appropriate frequency for the cell they are in.

The TBMAC protocol provides every mobile station with predictable access to the medium

with known probability and bounded delay[13]. TBMAC is an enabling technology, it can be configured to application specific requirements in terms of time bounds and probability in achieving them. Currently there is no proven time bounded MAC layers for infrastructure less, ad hoc networks. TBMAC provides an ad hoc wireless network with deterministic behaviour without requiring a master station.

3.1.1 Contention Free Period

In the contention period, TDMA MAC protocol rules are apply. Each slot is allocated to one and one station only. Stations which already have at least one slot allocated to them may piggyback further slot management requests on data packets.

3.1.2 Contention Period

This portion of the cycle is used only when a station is joining a cell. The contention period is divided into a number of slots. A station wishing to join an already active cell selects a random a number of these slots and transmits its request. Selecting a number of slots at random maximises the possibility that at least one of the requests will be successfully received by the other stations in the cell.

3.1.3 Hidden & Exposed Terminals

TBMAC relies on a cellular structure, the size of these cells is set such that all stations within a cell have full connectivity. This eliminates both the hidden and exposed terminal problems that hinder CSMA MAC protocols. By eliminating hidden and exposed terminals the provisions made to prevent or minimised there effects such as the RTS-CTS exchange are no longer necessary which in turn increases the portion of channel capacity available to useful traffic.

3.1.4 Slot Management

It is important to note front he outset TBMAC does not define an admissions control policy, slots are allocated on a first come, first served approach. Provision is made to allow an admissions control policy to be added later. All slot management functions, allocating, deallocating slots, notification of a stations intent to leave the cell are handled using Cristain's atomic broadcast protocol[10].

Bootstrapping

There are three possible scenarios in which a station may join a cell,

- Empty cell, only one station
- N stations enter an empty cell at the same time
- Station enters an already active cell

The TBMAC bootstrapping protocol has two distinct phases. A listening phase in which the station determines the current state of the cell and from which it selects the approximate bootstrapping procedure to follow. A full description is provided in [16].

3.2 Rtcomm, The Wireless Card Driver

The rtcomm wireless card driver is ongoing work. The driver is based on the open source driver for the Orinoco family of 802.11b wireless cards[1]. The wireless driver supports a range of wireless cards based on the Lucent Orinoco chip set.

RTAI real-time Linux[3] is used by the driver to provide a real-time deterministic interface to the wireless card. The deterministic behavour of this driver extends only to the software interface into the card. The packet queue between the Linux kernel and card is managed in real-time. The actual event of transferring the packet from the queue held in the linux kernel

[1]Source code available at http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Orinoco.html

to the wireless card remains non deterministic and may be effected by other ongoing events in the computer since the system bus in the computer only allows one data exchange at a time.

Chapter 4

Design

Implementing TBMAC requires a number of components which are, clock synchronisation, slot management and finally a TDMA MAC layer. Clock synchronisation is key. TDMA relies on tight clock synchronisation to function. The success of the clock synchronisation protocol will heavily influence the efficiency of the TDMA MAC layer. Once clock synchronisation is in place TDMA using a static set of slot allocations can be deployed. Following from static TDMA, TDMA with dynamic slot allocation can be provided by providing a slot manager that implements the TBMAC protocol. There are clearly three separate design concerns,

- Clock synchronisation
- TBMAC slot manager
- TDMA MAC layer

The key need is to separate concerns such that each component is separate and independent. This chapter will encompass the following, a section relating to each of the significant components, a section to discuss additions to the TBMAC packet header, a section showing how the individual components will link together to provide a working implementation of TBMAC. Finally an example is presented of how packets will be sent as well as how slot management is performed.

4.1 Clock Synchronisation

A prerequisite for TDMA is clock synchronisation. The performance and efficiency of the TDMA system is in part limited by the accuracy of the achieved clock synchronisation. Other factors such as interference may also impact efficiency. Accuracy to below 50μsec is desirable since the shortest permissible gap between 2 packets in 802.11b in ad hoc mode is the DIFS interval of 50μsec. By providing synchronisation to this accuracy the guard space between packets will be of the same order as is currently used by 802.11 in ad hoc mode. Exact synchronisation between clocks is not realistically possible owing to external factors such as physical propagation delays, stations are mobile, every 30cm distance between stations adds approximately 1 nsec to the propagation delay for example. Other factors such as non deterministic execution time when sending and receiving packets are also beyond our control. While the wireless driver (rtcomm) runs in real time and offers deterministic delays while interacting with the packet queue, there is still a possibility of delays while data is copied from the wireless card itself to or from these packet queues.

The rtcomm wireless card driver relies on RTAI real-time Linux to provide its real-time and deterministic guarantees in accessing the wireless interface. To provide a clock synchronisation service clearly two elements are required, a light weight and low impact way of altering the source of time information RTAI uses and a clock synchronisation service to provide agreement on the current clock value. During the design of these elements a key decision was to maintain as far as was practical full and total compatibility with all pre existing RTAI code, with the ultimate goal to provide clock synchronisation transparently to running real-time tasks such that existing code could take advantage of the benefits these new services provide without modification or even recompilation.

The decision to keep the clock manipulation functions separate from RTAI was twofold. Firstly by providing the bulk of the functionality in a separate module, significant modifications to the RTAI source code can be minimised and in particular introducing this class of functions into RTAI somewhat removes the transparency of the provided service. In addition

the clock manipulation module is independent of RTAI and as such could conceivably be used with other real-time Linux implementations or even as a replacement for the current combination of the operating system software clock and NTP as used in Linux.

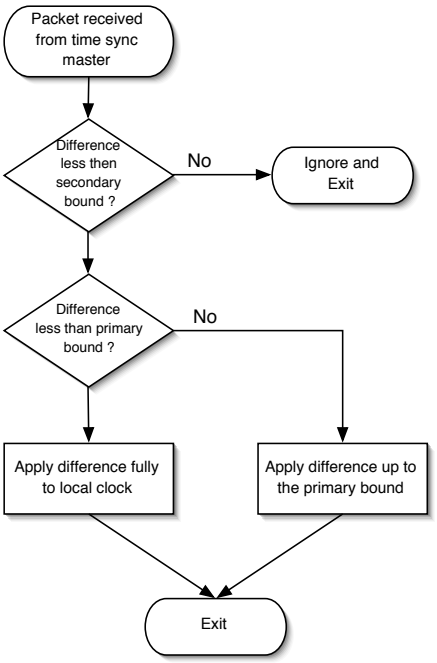

Fig. 4.1: Flow chart detailing the operation of the clock synchronisation algorithm

The key challenge is to provide a clock synchronisation service which is both flexible and lightweight. As outlined in chapter 2 the commonly used NTP protocol assumes that the client will have access to at least one peer or number of peers to synchronise with and that. This approach allows the local machine to learn and adapt and determine accurately the rate of clock drift and compensate accordingly.

A single station is defined as the clock synchronisation master. Other stations receiving packets from this station will determine, from an entry in the packet header, that the packet was sent by a clock synchronisation station. Stations synchronise to the clock value contained in these packets. The clock synchronisation station will be same station as the master station

if TDMA is implemented using a centralised slot management approach or where one station is involved in bootstrapping the other stations. Fully ad hoc synchronisation is not supported. A clock synchronisation master will always be required.

The proposed algorithm is stateless and consists of a two stage algorithm. If the difference between the local clock and the time stamp presented in a packet from a clock synchronisation master is within a defined bound the local clock is updated by that amount. If the difference is outside this primary bound but within a further larger secondary bound, the clock is adjusted by only the amount of the primary bound. If the difference is greater the secondary bound the clock is not adjusted. This approach ensures that the clock can drift to track the clock synchronisation station, but also ensures that rougue updates, updates contained in packets that where delayed due to contention on the medium are ignored. A detailed description of the algorithm are presented in fig. 4.1.

Clients of the clock synchronisation service bootstrap there clock values by listening for a packet from a clock synchronisation master and directly setting there local clock to the same value. Once synchronisation is achieved, periodic tasks may be started. From this point clock synchronisation continues as outlined in fig. 4.1

4.2 TBMAC slot management

This is the core of the system. The module provides a slot management solution which relies on a time bounded atomic broadcast protocol to achieve agreement. The key requirement of this module is to ensure that all slot management functions are carried out within the time bounds specified by the TBMAC protocol. The slot manager will execute the core TBMAC system consisting of the contention free period.

A design choice was to use an array to store this queue as to avoid the overhead of memory allocation and deallocation during processing of queued requests. All atomic broadcast messages are to be delivered at the beginning of the slot time before the packet (if there is one) is sent. This greatly simplifies the interactions within the system, in particular it avoids

a possible race condition between slot management functions and the TDMA system. This approach eliminates any possibility that some stations could run the slot management function and packet sending task in a different order, thus avoiding the creation of inconsistent views of slot allocation.

An additional requirement which is not addressed in the TBMAC protocol has been included in this implementation, to allow a station to request a specific global slot if for example it has specific timing requirements. This feature was added through the use of the `number_allocation` entry in the TBMAC packet header. The `number_allocation` entry will store the requested slot number.

4.3 TDMA MAC layer

TDMA encompasses the fundamental requirements of a time bounded deterministic MAC service. Clock synchronisation services will be provided transparently so the TDMA system can assume clock synchronisation is in place. Since clock synchronisation will be deployed at a level beneath RTAI the levels above will not be aware that there is clock synchronisation in place. However since the TDMA module through the rtcomm wireless driver copies the packets into the wireless card it is the last level before the wireless interface. The TDMA layer inserts the current time in nano seconds into the packet header immediately before it is sent to the wireless interface to be sent.

A key concern is to ensure that the TDMA system is not tightly coupled to the MAC protocol is the layer above, there is a need for communication which will take to form of three separate callbacks which simulate interrupts, one to notify the slot manager of the start of a slot, another to notify the slot manager that the TDMA system intends to transmit in a slot and finally a callback to notify the slot manager. The slot manager may choose to ignore some of these callbacks depending on the MAC layer that is in use.

All other interactions will use shared data through the services provided by the shared data module (sec. 4.5). While a packet is being transmitted the specific slot will be locked

to prevent the atomic broadcast queue or slot allocation state from being altered.

The MAC layer will be implemented in software above 802.11b, using RTAI real-time Linux to provide the necessary real time guarantees.

A array of pointers is maintained which represents each slot in the TDMA. If the contents of a slot are equal to the null literal the slot is not allocated to this station. This lack of distinction between ownership of a slot and implicit permission to transit in a slot ensures that the overhead in determining if a slot is allocated to the station is minimal and requires no more than a single comparison.

4.4 The TBMAC Packet

The current TBMAC packet does not make provision for the need for clock synchronisation services since it is assumed that synchronisation will be provided independently using GPS technology. In this implementation clock synchronisation is provided in a master slave fashion, thus provision was added to the TBMAC header to support these additional requirements. Four additional entries are added, two 64 bit time stamps, 8 bit message type entry and finally an 8 bit magic number entry is added at the head of the header to ensure quick detection of a valid packet. The modified TBMAC header is shown in fig. 4.2.

4.5 The Components

The system will be made of a number of individual modules each with a single specific role. A diagram detailing the relationships is provided in fig. 4.3. Each module is discussed in turn below,

Shared Data Module

This module is a central point for all data structures which are shared with other modules. The TDMA table and slot bitmap will be stored here. This modules will also provide services to manage concurrent access to these data structures to ensure consistency is maintained.

```
typedef struct{
        unsigned magic_id : 8;
        unsigned long long source_address : 48;
        unsigned long long dest_address : 48;
        unsigned int slot_bitmap; : 32
        unsigned seq : 16;
        unsigned slot_number : 4;
        unsigned message_type : 1;
        unsigned msg_type : 8;
        AdditionalInfo additional_info[2];
        unsigned long long snd_time;
        unsigned long long rcv_time;
        unsigned char contents[512];
}cfp_message_hdr;
```

Fig. 4.2: Modified TBMAC Packet

Dynamic TDMA Module

This module handles all incoming and out going data which passes through the wireless interface. The module tightly integrates with the rtcomm driver module. The dynamic TDMA module is oblivious to the other modules and transmits in any slot in which it has data to transmit. It performs no management of the TDMA slot table and is independent of slot management protocol that is in use. It will however notify the slot manager through the use of callbacks of events such as receipt of a data packet and the imminent transmission of a packet in the current slot.

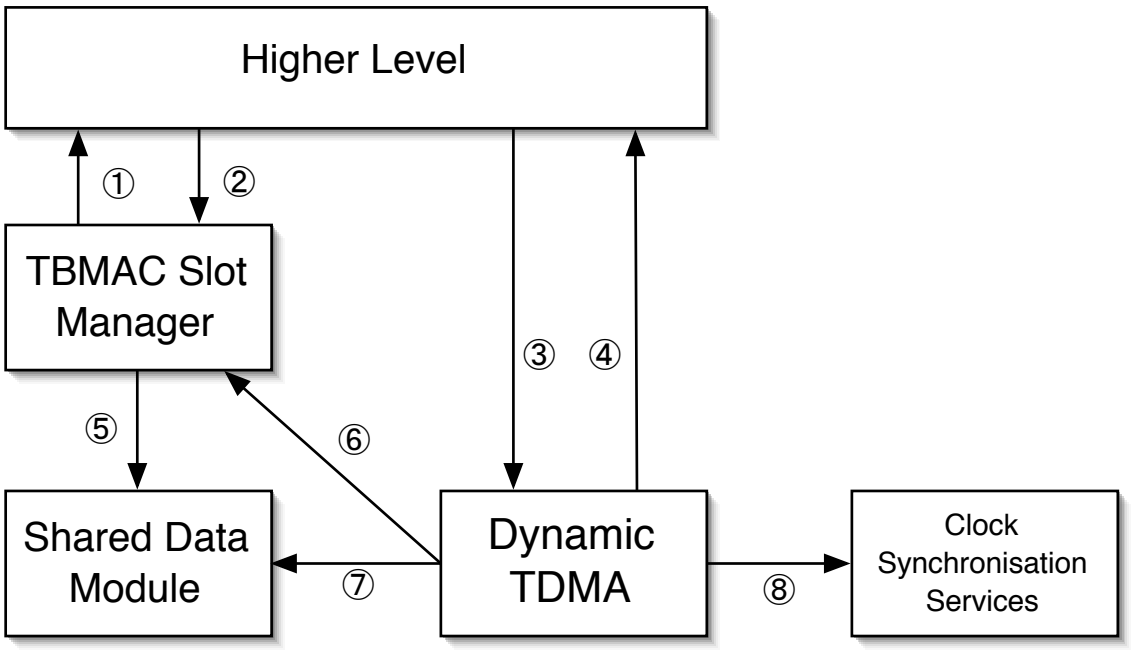

Fig. 4.3: High level view of module relationships

TBMAC Slot Manager

This is the slot manager module which manages the allocation and deallocation of slots in addition to admission control. The module handles all requests from the higher level to perform allocation, deallocation and leave services as defined by the TBMAC protocol. It interacts with the TDMA system through protected shared data structures provided by the shared data module.

Higher Level Module

This a simplified implementation of the higher layer that is using the time bounded service provided by the combination of the TBMAC slot manager and the dynamic TDMA module. It will provide an interface into user space allowing bidirectional interaction through the use of FIFOs. This higher level module will form a significant part of the evaluation of the system

since it will maintain the current state and result of slot allocation and deallocation requests.

Clock Synchronisation Services

This modules provides an interface into the `rdtsc` instruction used by RTAI to maintain its internal clock. Functions are provided to manipulate and modify this so that all stations see the same time. Relative time is used only.

4.5.1 Inter Module Relationships

This section will detail the precise relationships and interactions between the various modules that make up the system. Fig 4.3 shows the relationships numbered below.

1. A callback to notify of the successful or unsuccessful allocation or deallocation of a slot following a request.

2. Interface into the slot management system, functions such as allocate slot, deallocate slot, leave, allocate specific slot, allocated slot count. A function is also provided to initialize the callbacks discussed above.

3. Provides 3 functions, the ability to send a packet and to send a packet repeatedly, in addition it allows the higher level to register a callback for the notification of receipt of packet.

4. Using the callback registered above the TDMA module may notify the higher level of a received data packet.

5. The slot manager may manipulate data in the shared module.

6. The TDMA module notifies the slot manager of the arrival of a packet, or the imminent transmission of one

7. The dynamic TDMA module may manipulate data in the shared module.

8. Each packet contains a time stamp, the difference between this time stamp and the current local time is passed to the clock synchronisation module.

4.6 Sending A Packet

There is clear need to provide separation between the TDMA system and the higher levels. Fig. 4.4 shows the steps required to send a packet. On the right is the interaction between the user application and the slot management service. The left side shows the TDMA system running in parallel. Consistency is maintained through the shared data module which ensures the TDMA table is locked while it is in use.

4.6.1 Real Time Task

This is the core of the TDMA system. It is invoked by RTAI at each slot time. It first carries out slot management tasks such as slot allocation and deallocation and the management of the atomic broadcast queue, referred to as ATB in the figure.

If the slot is allocated to this station the slot manager is notified. If the slot manager is TBMAC this notification is used to set the delivery time on all new pending atomic broadcast requests. Using the reference in the TDMA table as a pointer to a packet that packet passed to the rtcomm driver which copies the packet to the wireless card.

4.6.2 Send Packet

An api call to allow a user application to send a packet will be made available in the form of `send_pkt(int slot_number,struct message *msg);`. To send a packet a pointer to the packet must be placed in the TDMA table. First the local slot number is converted into a global slot number and if the slot is currently allocated this station an entry is made in the TDMA table pointing to the packet.

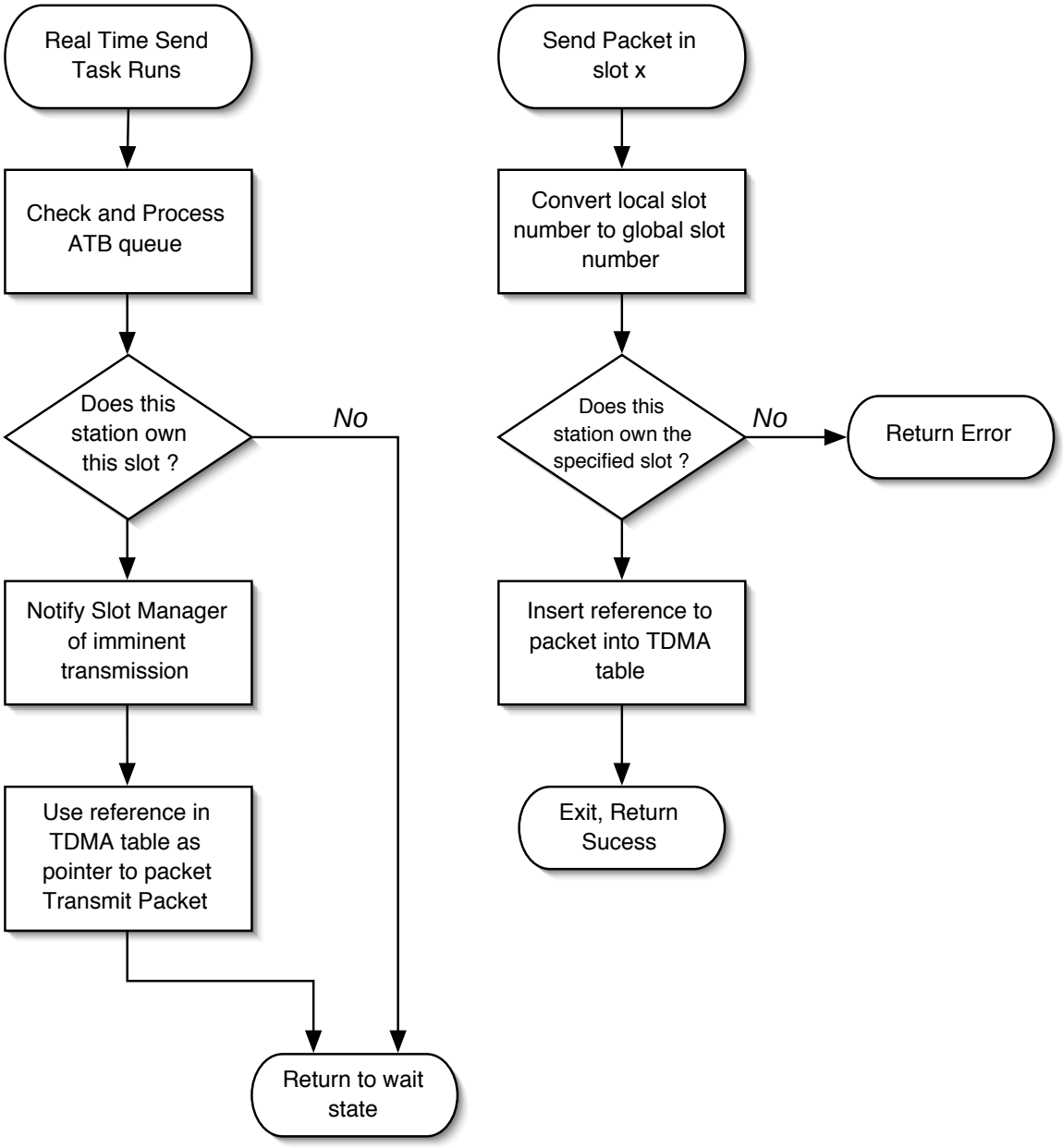

Fig. 4.4: Flow chart detailing how packets are sent

4.7 Slot Management

Again there is a clear separation between the receipt of slot management requests and there application to the current slot allocation state.

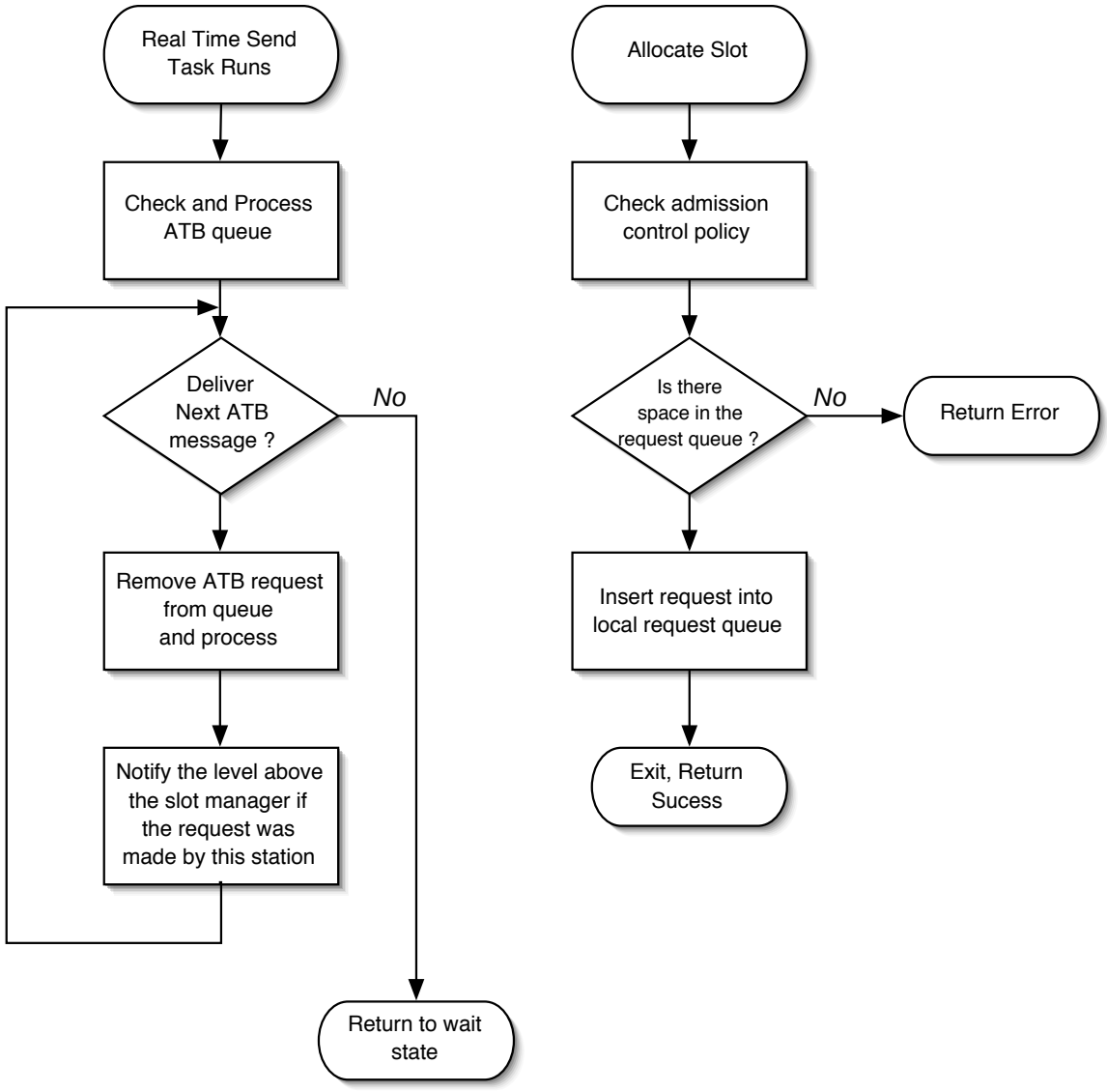

Fig. 4.5: Flow chart detailing how slots are allocated

4.7.1 Slot Management

Slot management is carried out at the start of every slot time, before a packet is sent. Slot management is performed regardless of whether the slot is allocated to that specific station or not. The current atomic broadcast queue is checked for any requests whose delivery time has arrived. Slots are allocated and deallocated where possible as dictated by these requests. If the request was made by this station the level above the slot manager is notified of the outcome of that request. If the slot is allocated to this station any slot management requests currently queued are time stamped for delivery and are added to the atomic broadcast list. This list is copied into the headers of all packets sent.

4.7.2 Allocate Slot

An api call to allow a user application to send packets will be made available in the form of `int allocate_slot(void);`. Functions will be provided to allocate a specific global slot as well as matching functions to deallocate one specific slot or to deallocate all slots currently allocated. To request a slot the level above the slot manager invokes either the `allocate_slot` or `allocate_specific_slot(int global_slot_number)` functions. The request is then queued. To allocate a slot, the admissions policy is queried first then the request is queued with all new slot management requests which in due course will be time stamped and sent.

4.8 Summary

The design of the system is highly modular and all the important concerns are separated into individual modules. The critical slot management service is fully independent of the TDMA module and is fully interchangeable thus allowing different TDMA MAC layers to be used. This flexibility is key to the development and testing of the finished system. The design of TBMAC has been slightly modified to allow for the allocate specific slot function.

A critical design decision was to provide clock synchronisation externally to RTAI thus maintaining transparency. A further decision was the adoption of the processor cycle clock,

the tsc as the basis for time keeping in the system, this ensured low level compatibility with RTAI and also provides a source of high resolution clock information. The chosen approach to integrating clock synchronisation relies on the hardware abstraction layer used by RTAI and as such can be deployed on all platforms RTAI currently support that provide tsc like functionality.

Chapter 5

Implementation

This chapter will discuss the finer details of the solution. In particular each of the components of the proposed system are discussed. An overview of the layout of the full system is presented in fig. 5.1.

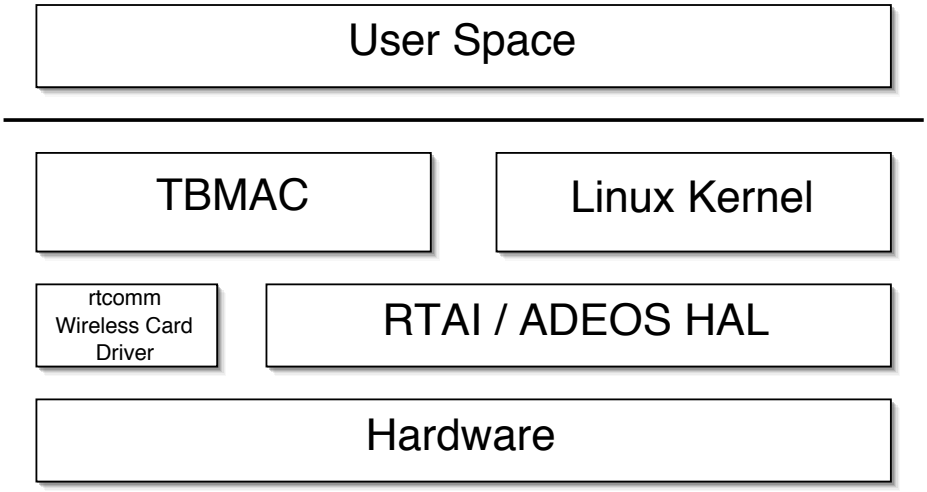

Fig. 5.1: High level view of component relationships

5.1 Real Time

Developing code to operate in a real-time environment is extremely challenging particularly when the code is interacting with external events such as the receipt or transmission of packets through a wireless card. Debugging in real-time under these conditions is not realistically possible, since external events occur at such frequency. While the core elements of logic can be extensively tested outside real-time, the interaction between system elements and in particular between the code and the underlying physical hardware cannot leading to significant challenges in testing and debugging.

A failure or error of any kind typically leads to a kernel panic which means tracing the root cause of an error is an extremely arduous process. These concerns lead to the use of the incremental implementation approach. Each component is tested and proven before moving on, by refactoring known good code the system evolves into the next phase of development.

As shown in fig. 5.1, the solution is implemented entirely in kernel space and in real-time. The RTAI/ADEOS layer sits directly between linux and the hardware, its primary purpose is to intercept and manage interrupts such that real-time tasks can meet there timing deadlines. The lowest priority real-time task will always have priority over all linux processes including the kernel.

This solution has been deployed only for Pentium processors but can be applied to other modern processors families such as the PowerPC frequently used in embedded systems. Many modern processors contain a CPU clock consisting of a simple counter which increments on every clock cycle. This is commonly known as the time stamp clock, or tsc. This counter is typically represented as a 64 bit wide register and is directly accessible via a special assembly language instruction. There is no software or operating system interference, a 1Ghz clock speed gives a resolution of 1ns.

Following analysis of the source code of RTAI and a review of the capability of the Pentium family of processors the following key observations where made,

- There is no reasonable possibility of the tsc counter wrapping around[1].
- The `rdtsc` instruction is read only.
- The `rdtsc` instruction is executed entirely in hardware, it is not effected by interrupts.
- In RTAI all accesses to the tsc or equivalent are made through exactly one (well known) function.
- When running in one shot mode all timing in RTAI relies on the result of the `rdtsc` function.

5.2 RTAI clock manipulation

Currently there is no provision in RTAI to support distributed time coordinated applications. Additions to RTAI such as RTnet[4] provide high level services to support limited co-ordination across broadcast type networks, specifically ethernet. A master slave relationship is used and all devices must be registered with the master before data exchange can take place. While a TDMA structure is used it is neither dynamic nor is it fault tolerant, making it unsuitable for this application.

To address this lack of functionality a RTAI clock manipulation module was devised. It introduces a small amount of extra functionality into RTAI, when running in one shot mode[2] RTAI relies entirely on the `rdtsc` instruction (or simulation of) to determine timing. In one shot mode a task is rescheduled on each execution in doing so it uses the clock value at that time. RTAI uses a hardware abstraction layer so that it can run on different hardware platforms this isolates all architecture specific code into its own group of header files.

All calls to the CPU time stamp counter are made through the function `rdtsc()` which is contained in the `rtai_hal.h` header file. This header file is always included in all modules using RTAI functionality, thus by modifying this function alone it possible to effect the view of time seen by all parts of RTAI, and in doing so to effect the scheduling and timing decisions

[1] At 1Ghz it would take 548 years before the counter reaches its maximum value

[2] This is the recommended mode of operation and offers the highest level of precision in the timing of the execution of tasks

made by RTAI. Clearly extreme caution must be exercised, any sudden large changes in perceived value of the `rdtsc` function may lead to unpredictable behavour, and in particular failure to execute tasks, multiple execution or worse still a crash.

5.2.1 Alterations to RTAI

Clearly the source code of RTAI needs modifications to expose the necessary functions and variables to allow for clock synchronisation. Only three simple modifications are required to insert support for clock synchronisation in RTAI. The aim is to make the absolute minimum changes.

The `rtai_hal.h` header file requires 2 modifications. The `rdtsc` function returns directly the 64 bit value of the tsc register on the processor, this solution simply adds or subtracts as appropriate a 64-bit value from the tsc value before the function returns, in doing so at this low level RTAI is completely unaware of the alteration. Since this function is the basis of all time keeping in RTAI adjusting the value within by adding or subtracting from the tsc value effects the scheduling of all tasks in RTAI. A shared variable named `TSC_OFFSET` defined as `extern` type is also required so that all RTAI modules see the same offset and that the same offset is applied to all RTAI modules. The modified `rdtsc` function is presented in fig. 5.2

An additional kernel module is required to provide functions to manipulate the value of the shared variable `TSC_OFFSET`. This module also creates and initialises `TSC_OFFSET`. This module must load prior to RTAI, configuring linux such that the module is loaded during system startup avoids difficulties later on.

It is important to note that these modification in no way effect the performance or functionality of the RTAI distribution. Other than adding an offset no further work is done within RTAI, any modules complied against the unmodified version of the header files will continue to function normally even if the clock manipulation functions are used. There is only one cause for concern, if a module directly calls `rdtsc`, `rdtsc()` is not a documented function, and is known by different names on different hardware platforms. Programmers should use the `rt_get_count()` instead which gives the same result but using a function exported by

```
static inline unsigned long long arti_rdtsc (void) {
    unsigned long long t;
    __asm__ __volatile__( "rdtsc" : "=A" (t));
    return t + TSC_OFFSET;
}
```

Fig. 5.2: Adjustment to RTAI code

RTAI in the kernel. Using `rdtsc` will lead to a disparity in the rdtsc values seen leading to failure. Logically it is important to remember that the tsc cannot be negative, it will always be possible to add or subtract a value from the tsc such that it will remain positive as it is synchronised with another processors tsc.

Clearly if there are no currently running periodic tasks it is conceivable to make sudden large alterations to the apparent tsc value the fact that time is monotonically increasing thus time can never go backwards may be ignored safely. The tsc is read only and is accessed frequently from within RTAI itself thus embedding logic inside these requests to perform synchronisation would lead to unacceptable overhead.

5.3 Clock Synchronisation Service

Traditional clock synchronisation protocols, NTP[2], Cristian[9] and Berkeley[19] require multiple message exchanges between at least two stations, multiple messages are required to get an estimate of the round trip time, this introduces what I will refer to as the chicken and egg scenario.

To implement TDMA tight clock synchronisation is required, but to transmit a packet (to perform clock synchronisation) using a TDMA type MAC layer requires that a TDMA schedule be in place, that there already exists sufficient synchronisation to use TDMA. Multiple message exchanges are not acceptable since they are in addition to the agreement requirement

of the TBMAC protocol thus further increase the time bound to sending the first data packet. It must therefore be possible for a station to gain clock synchronisation without transmitting itself, this makes current protocols unsuitable.

Clock synchronisation was implemented in a separate module. The module is lightweight and can be loaded into the kernel without having to load RTAI first. Two functions are exported. One `int adjust_tsc(signed long long delta)` is the primary function used which implements the protocol as shown in fig. 4.1 earlier. A second function `fast_adjust_tsc` is provided which allows an unrestricted alteration of the value of the return value of the `rdtsc` function. This function is used only during the bootstrapping phase to eliminate the need for time spent awaiting convergence. This approach ensures that the clock synchronisation protocol in no way extends the current bound TBMAC defines for

5.4 Shared Data Module

RTAI provides extensive synchronisation support through the use of semaphores. In order to maximise performance and minimise the delays caused by waiting for locks, locking is applied at the global slot level, thus only the slot currently in use is locked. This provision eliminates a possible race condition between the TDMA module sending a packet and the higher level passing a packet to be sent. A second semaphore is provided to secure access to the atomic broadcast queue.

The shared data module is the home of all key data structure used in the system. The TDMA slot table consists of an array of pointers to packets. In order to support the `send_packet_repeated()` function call, an additional integer array is required to log the status of a packet, is it to be retained for retransmission or should it be removed from the TDMA table when it is sent. The global atomic broadcast queue is also stored here.

5.5 The slot manager

The slot management module implements the TBMAC protocol itself and is independent of the TDMA module, all interaction taking place via the shared data module.

The slot manager maintains the slot bitmap. This stores in two bits the current state of each slot, `AVAILABLE`, `COLLISION`, `OTHER` or `OWNER`. Though not used in this implementation in a fully dynamic implementation the slot bitmap is the corner stone in enabling the joining station to learn about the current slot allocation state. The slot bitmap is stored as a 32bit integer allowing the system to be configured for up to 16 slots. It is possible to support up to 32 slots through the use of a 64bit integer.

Slots are allocated on a basis of the lowest available slot number available first, while simple this is the most effective approach, however this approach is overridden if the allocate specific slot function is use at which point the value of the `number_alloction` entry in the TBMAC packet header is used as the slot number.

5.6 The TDMA module

Owing to the dynamic allocation of slots the implementation of the TDMA module is not as straight forward as could be expected. There are three distinct approaches to the management of the periodic packet sending task within the TDMA module.

- Reschedule the task following each execution so that it will execute again to transmit in the next currently allocated slot.
- Have an individual task for each slot and enable/disable as required.
- Have one task which executes regardless of the allocation state of a slot, the task confirms the slot ownership and if favorable transmits.

While the first option presents the most resource efficient implementation it presents some significant difficulties in implementation. The act of rescheduling and the overhead in calculating the next execution time is significant. A major issue is that the scheduling decision

is made based on the state of slot allocations at a time before the next (expected) slot, in due course a slot may be allocated or deallocated before this expected next slot leading again to further rescheduling. Since this system is intended to be hard real-time this approach would lead to significant variances in the length of time a task takes to execute.

The second approach avoids the rescheduling issue of the first approach but is resource hungry with a requirement for one task per slot. The act of starting and stopping tasks in accordance with slot allocation and deallocation events may lead to timing failures. The message must be first placed into the slot in question and the task restarted.

Both of these approaches suffer from the overhead of managing the task, the third approach allows for a single task executing in every single TDMA transmission slot. In doing so there is no overhead. The task checks to see if the slot is allocated or not, this can be implemented simply by checking if an entry points to a memory address or is equal to NULL. Implemented in this form, allocating and deallocating a slot becomes an atomic process from the perspective of the TDMA process. The act of placing a pointer to a piece of memory implies permission to transmit the contents of that piece of memory in that slot. In addition since RTAI will be operating in one shot mode by the task executing in every slot it will be rescheduled following each execution, its next execution time will reflect the current clock i.e. the most recently synchronised value, the other two approaches suffer from up to a full cycle latency before they reflect the most up to date clock value.

There are no conflicts since clocks are synchronised across all stations, once the task has determined that it has not got permission to transmit, in parallel the station which has permission to transmit has just determined that it has permission and begins copying the packet into the cards buffers so there is no possibility under normal operation that the higher priority send task would be running when a packet arrives[3]. While from a resource utilisation this approach may seem greedy, the station would have been idle in face of not owning the slot, in addition under the worst case load hypothesis one station could possibly own all slots.

[3]The read task runs at a lower priority

A key requirement is to keep the atomic broadcasts ordered by time. since a broadcast message may be sorted using three criteria, its delivery time, the requesting stations MAC address and the requests sequence number. Instead of explicitly sorting the queue upon each addition or group of additions, such an approach is extremely time consuming and would require three passes to ensure the queue is in correct order, one to sort by time, a further one to ensure if two or more requests have the same delivery time and to resort them appropriately and a final pass to ensure that duplicates are removed and any requests with the same delivery time and requesting mac address are ordered by request sequence number.

Instead each request is inserted in the correct location in the queue when it is added thus ensuring the queue always remains in correct order without the need for an explicit recourse to a sort algorithm. A constrained linked list is provided thus allowing rapid addition and removal of entries without the need to allocate or deallocate memory. When updating the order following addition or removal only the pointers need to be updated, there is no need to copy entries.

Fig. 5.3 shows an example scenario of slot allocations and shows the relationship between the local slot table and the global slot table as well as how the NULL and data packets relate to the TDMA slot table. A three level system is used, the local slot table provides a level of indirection into the global slot table. If an entry (in the local slot table) is equal to -1 the entry is unused and does not point to a valid global slot. This structure is designed such that there is minimal overhead in determining the state of a slot, if the global slot entry is NULL[4], the slot is not allocated to this station, otherwise the entry is a pointer to a piece of memory. There are three types of entry in the global slot table,

- A pointer to a normal data packet
- A pointer to a special NULL packet
- The NULL pointer

The NULL pointer signifies that the slot is not allocated to this station and data may not be transmitted in the slot it refers to.

[4]The NULL literal is represented by 0 in the diagram

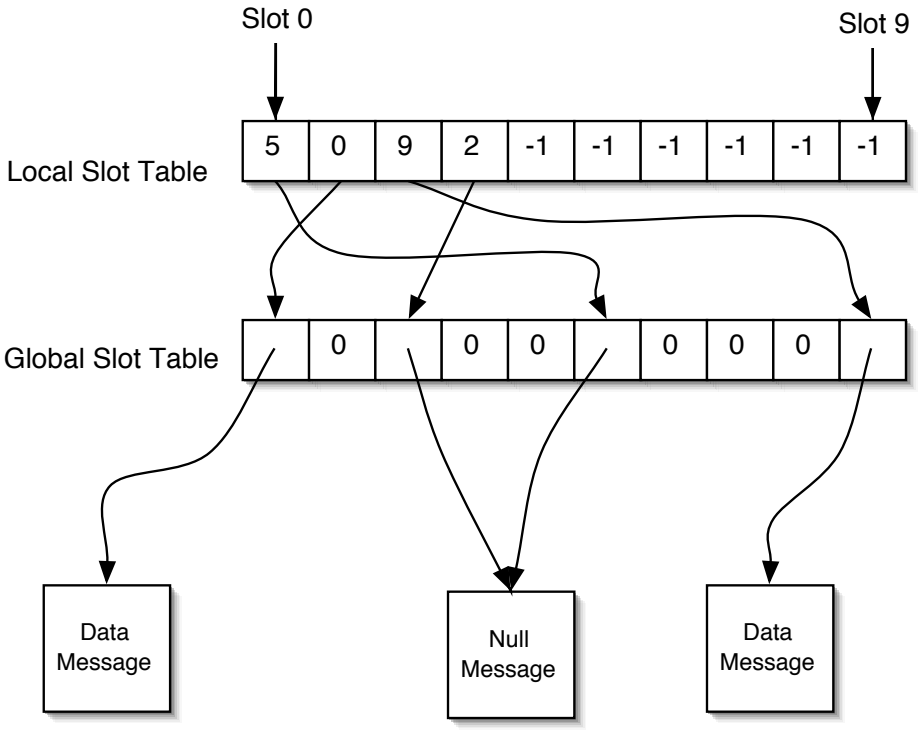

Fig. 5.3: Send Packet

The special NULL packet is used whenever a slot is allocated to a station but in which there is no data to transmit, this NULL packet can potentially be shared amongst many slots and carries no data apart from the TBMAC header. When a data packet is sent a slot enters this state, thus an allocated slot will always transmit a packet, this is a key requirement of the atomic broadcast protocol used by TBMAC.

Otherwise the entry in the global slot table is a pointer to a packet which is copied to the wireless card. A packet may be marked as periodic such that when transmitted the pointer is not replaced by the NULL packet instead it is left in place this facilitates periodic traffic.

This approach fully de couples the slot management functionality from the actual TDMA system. There is no need to query ownership of a slot, the presence of data to transmit in a slot implies ownership and permission to transmit in that slot. The slot manager ensures

that consistency is maintained so the TDMA system does not need to have any knowledge of the slot management procedure in place.

The key thing to note is that they are all pointers which means the packets themselves are not copied, this leads to the ability to efficiently support periodic traffic for instance telemetry or periodic sensor data. Since the process that created the packet still retains a reference to the packet it may alter its contents.

5.7 Summary

Clock synchronisation has been implemented with extremely encouraging results. The approach taken to clock synchronisation has ensured that a consistent clock is seen by all elements of the TBMAC system as well as the underlying real time linux system.

The implementation is fully modular and allows the slot manager to be interchanged with few if any modifications to the TDMA system.

Chapter 6

Evaluation

This chapter will outline the experiments carried out, there goals, results and conclusions that can be drawn from them, firstly the equipment used and configuration is outlined.

6.1 Test Setup

All experiments where conducted using identical equipment with the same configuration,

- 3 Dell Latitude C400 laptops, 730MHz Pentium 3 Processor, 256 Mb RAM
- Redhat 7.3 and RTAI 3.0
- All laptops ran a 2.4.20 kernel patched by RTAI
- All power management functionality was disabled in both the kernel and system BIOS
- Lucent Orinoco Gold 802.11b PCMICA wireless cards

The laptops where arranged in a circular formation, antennas where positioned such that there where all an equal distance apart. All experiments where conducted at 2Mbit/sec to ensure the effects of local interference was minimised, 802.11 channel 2 was used, this frequency was not in use by any other 802.11 devices in the locality. All machines where allowed to warm up for a short while before commencement of experiments. Two very different environments where chosen to conduct the experiments, these environments where

chosen to show two extremes in which an 802.11 device may be used.

Location 1

In a first floor room in the O'Reilly Institute, Trinity College Dublin, home of the Dept of Computer Science. This represents a extremely aggressive wireless environment with many other 802.11 sources, from both infrastructure based and ad hoc networks. Significant other possible sources of interference are from electrical systems, mobile phones, computer systems and bluetooth devices. An electric railway also passes within 20m of the location[1]. This environment represents in the authors opinion the worst practical scenario that an 802.11 device may operate in.

Location 2

In the authors garden, this represents the best practically achievable environment, with no other active 802.11 devices, mobile phones or electrical systems in close proximity of the 802.11 cards used to perform the test. This is not a clinically clean environment but represents the best case that is practically achievable, a perfect interference free environment is not in the authors opinion of any use when presenting work evaluating the real world performance of 802.l1.

6.2 Experiments

A series of experiments where carried out to evaluate the various components that make up the TBMAC implementation. The experiments where broken down into 4 separate areas.

- Determination of the propagation delay
- CSMA performance

[1]Recent work presented in [36] shows significant issues with relation to modern, inverter (GTO/IGBT) driven trains, which are present at this location. Unfortunately the work concentrated on frequencies below 1Ghz (802.11b operates at 2.4Ghz), frequencies above 1GHz are not covered by relevant European rail industry standard, EN 50121

- TDMA performance
- Timebounds in TBMAC

6.3 Determination of the propagation delay

Clock synchronisation relies on the fact that in a TDMA environment collisions are rare, so the propagation time of a packet from sender to receiver should be constant, however the propagation delay will vary dependant on the packet size. Before clock synchronisation could be deployed it was necessary to estimate actual propagation time in order to correct the received time stamp for the expected arrival on the remote station.

The propagation delay was calculated by a periodic task on one station sending a packet padded to the appropriate length containing the estimated delivery time on the remote station, the remote station updates its clock to this value using the clock synchronisation protocol detailed earlier in chapter 4.1. From the rate and size of clock updates the accuracy of the propagation estimate can be determined.

The estimated delay was adjusted until the median average adjustment to the local clock required to maintain clock synchronisation is close to 0, further minor adjustments are made to reduce the variance in updates. Owing to the variability in the rate of clock drift an exact value of the propagation cannot be calculated, and indeed is undesirable since it would over fit the value making it inflexible.

An estimate sufficiently accurate to maintain clock synchronisation within the required bounds is sufficient. As shown in fig. 6.1 on the following page the relationship between propagation delay and packet size is not a perfectly linear. This may be attributed to the flexibility in the choice of the propagation delay and also the fact memory on the wireless card during packet transmission is allocated in discrete blocks, the time to allocate a single block remains constant regardless of whether a single byte of a allocate block is used or the entire space, this could explain the three clear steps in the graph shown in fig. 6.1.

A sample plot of the updates applied to the clock on station 2 may be found in fig.

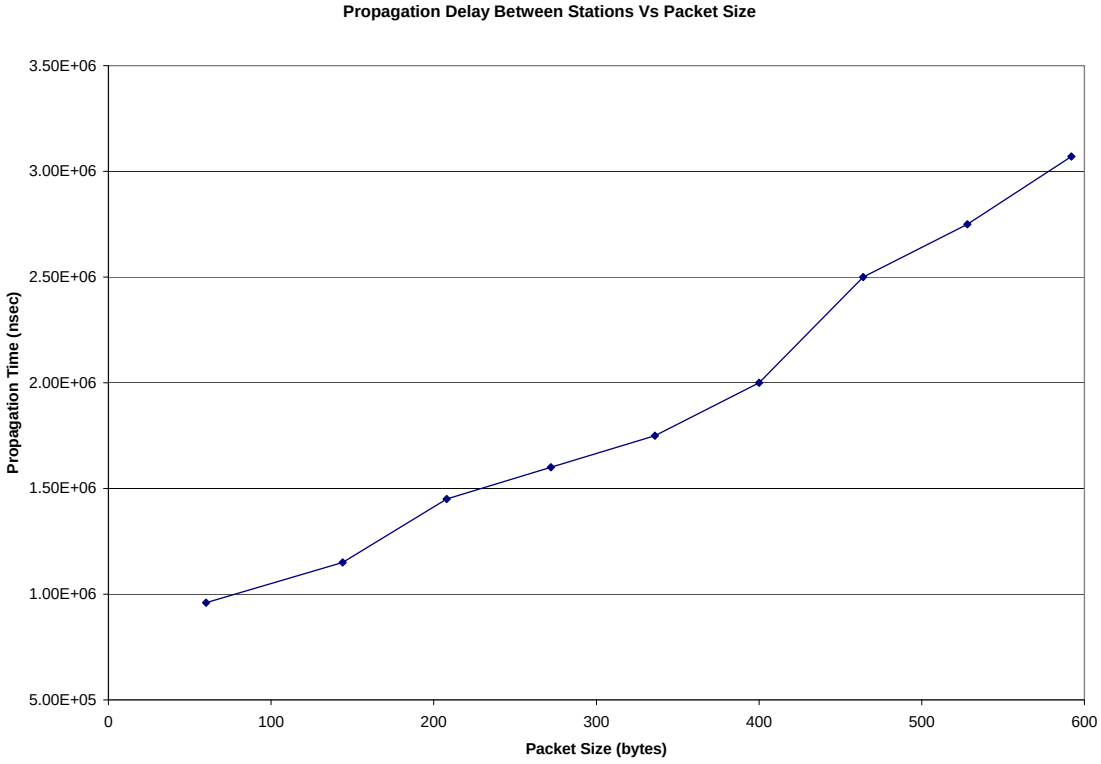

Fig. 6.1: Plot of the calculated propagation delays over a range of packet sizes

6.2. There is clearly oscillation between the upper and lower bounds, this is unavoidable. It is impossible to decern between an update that was slightly delayed due to actual clock drift or due to the variance in the execution of a real time task or equally due to minor variances in processing delays. In order to cope with the fact processor speeds are not identical across stations the tsc value is converted to nano seconds by dividing by the processor speed measured in Ghz, each time stamp must be converted to nano seconds on the transmitting station and converted to local tsc units on the receiving station. Processor speed is calculated only to the nearest KHz at boot time, processor speed varies slightly in due course leading again to further errors.

A brief analysis of the updates to the clock on one station is presented in table 6.1.

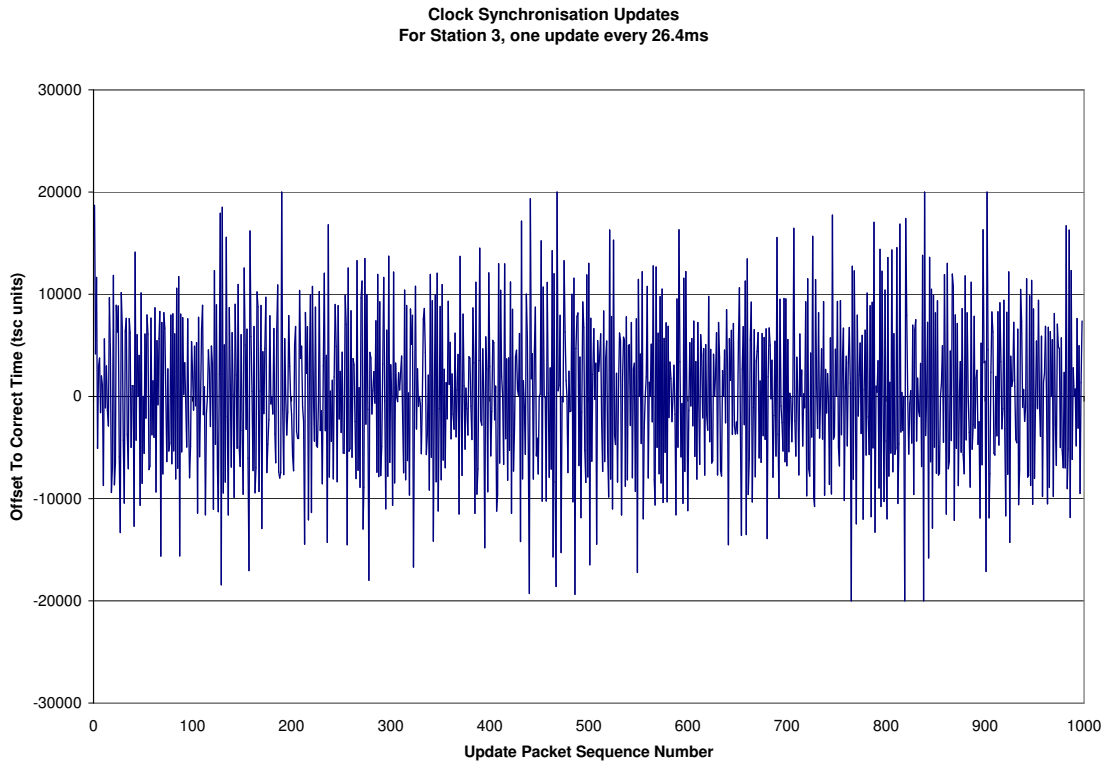

Fig. 6.2: Plot of the updates made to the local clock.

All clock synchronisation information presented here was gathered while the full TDMA system was running, transmitting and receiving data and thus provides accurate data of the performance of the clock synchronisation protocol under the exact conditions it would operate under. These results show that the average update to the clock is on average less than 1μsec.

RTnet[4] an extension of RTAI to provide real-time ethernet support also implements a clock synchronisation protocol with similar requirements to those detailed in chapter 3. They share a number of similarities in there use of broadcast packets. Compared to similar work detailed in[20], this protocol offers significant benefits, for example in[20] convergence requires several minutes and is subject to occasional spikes in excess of 100μsec, using the protocol implemented here convergence is virtually instant, this an essential requirement to

Update Frequency	Median adjustment	Average adjustment	RMS error
28.0ms	-79.4ns	4.7ns	12895ns
27.2ms	-28.8ns	-58.3ns	16439ns
26.8ms	-619.2ns	-122ns	11695ns
26.4ms	-223.3ns	-17.8ns	11531ns
26.0ms	-606.8ns	-107.3ns	10987ns

Table 6.1: Analysis of updates to the clock on station 2

enable a station to gain access to the medium within a bounded time. The performance from the clock synchronisation protocol as detailed in table. 6.1 shows improved stability and a tighter level of synchronisation. The average update is approximately zero, the RMS value shows a more accurate indication of the stability of the clock synchronisation protocol, since positive and negative updates don't cancel each other out. The RMS value varies between 10 and 16μsec.

6.4 CSMA experiments

The aims of this set of experiments was to evaluate the performance of 802.11 in the face of collisions and contention. These results in the form of round trip times are compared to results of the same experiments but using a TDMA MAC layer in chapter 6.5. The overall aim of these experiments is to show that TDMA offers stable time bounded contention free delivery even when the medium is heavily loaded.

All experiments consisted of a packet containing a 512byte payload as well as containing source and destination addresses as well as time stamps and sequence numbers required to record the experiments results. The CSMA packet is a total of 548 bytes in length, its contents are shown in fig. 6.3

Each experiment consisted of the master sending 1000 packets to N stations. These stations will upon receipt of a packet (from the master) reply to the master thus producing a

```
typedef struct{
    unsigned magic_id                   : 8;
    unsigned long long source_address   : 48;
    unsigned long long dest_address     : 48;
    unsigned seq                        : 16;
    unsigned msg_type                   : 8;
    unsigned long long snd_time;
    unsigned long long rcv_time;
    unsigned char contents[512];
}message_hdr_csma;
```

Fig. 6.3: CSMA packet contents

total of 1000 * (N + 1) packets. All transmission took place using standard 802.11 broadcast packets, no RTS/CTS or ACK packets where sent. Under the CSMA MAC protocol all stations should attempt to reply instantly. In a simple two station scenario, one master one slave contention is entirely a function of the slave contenting with the master to transmit, that occurs only when the master transmits packets at a faster rate than the slave can reply. For more than 1 slave contention is primarily between the slaves and to a lesser extend with the master.

The round trip times presented indicated a best case time of approximately 6.9ms from the time the request to send the packet is made, until the packet is fully retrieved from the card following the round trip. All time stamps are generated using the Pentium processors time stamping clock (tsc) which offers a resolution of approximately 1.2nsec.

In CSMA mode all packets where generated and sent from a periodic task at set intervals, the slave station upon receipt of a packet changed the originators address and instantly sent the packet back, other the changing the originators address the slave carries out no other processing. All the slave stations should attempt to reply at the same time.

6.4.1 CSMA Round Trip Results

Fig 6.5 & fig 6.4 show clearly the non deterministic and random behavour of the CSMA MAC protocol, the minimum round trip time is between 6.9 msec and 7.0 msec.

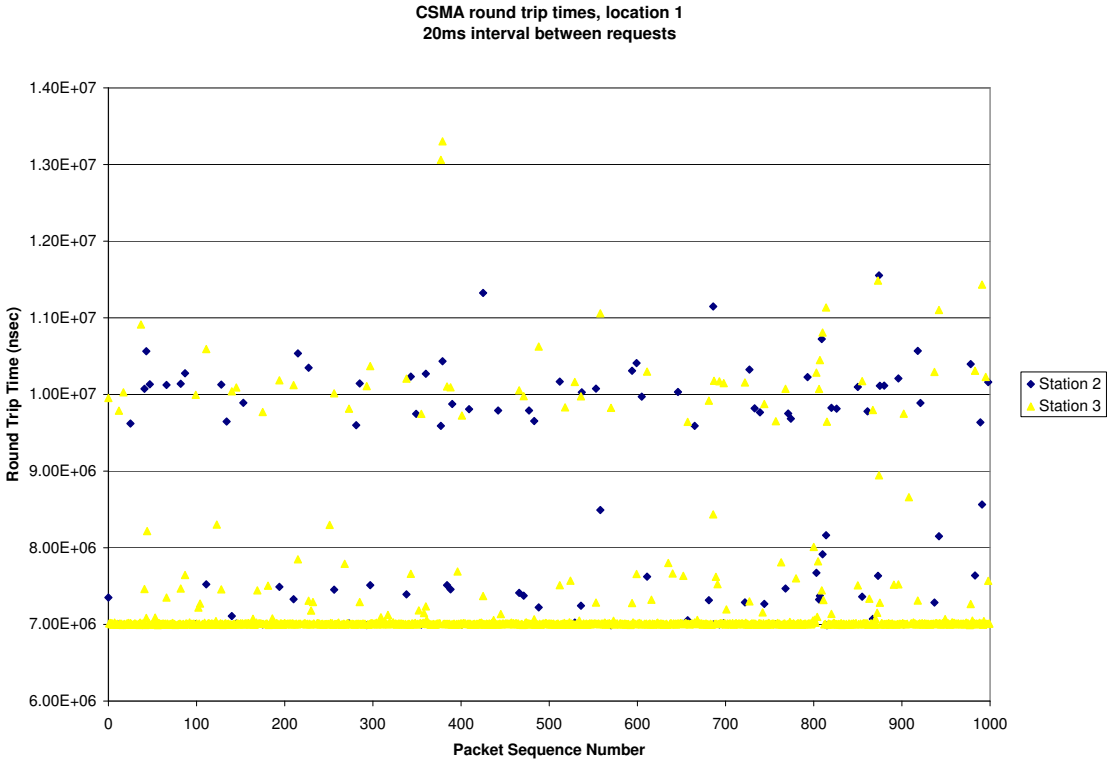

Fig. 6.4: Plot of heavy contention using CSMA at location 1

As detailed in table. 6.2 there is an extremely high level of packet loss. Clearly in these experiments station 3 has some advantage over station 2. Since the station are all equidistant from each other during all experiments propagation effects have been eliminated. Further investigation of the kernel logs over many experiments on both laptops revealed that the estimated clock speed of station 3 varied relative to station 2 by up to 11,000Hz in favour of station 3. This advantage is fractional but is more than sufficient to give station 3 an

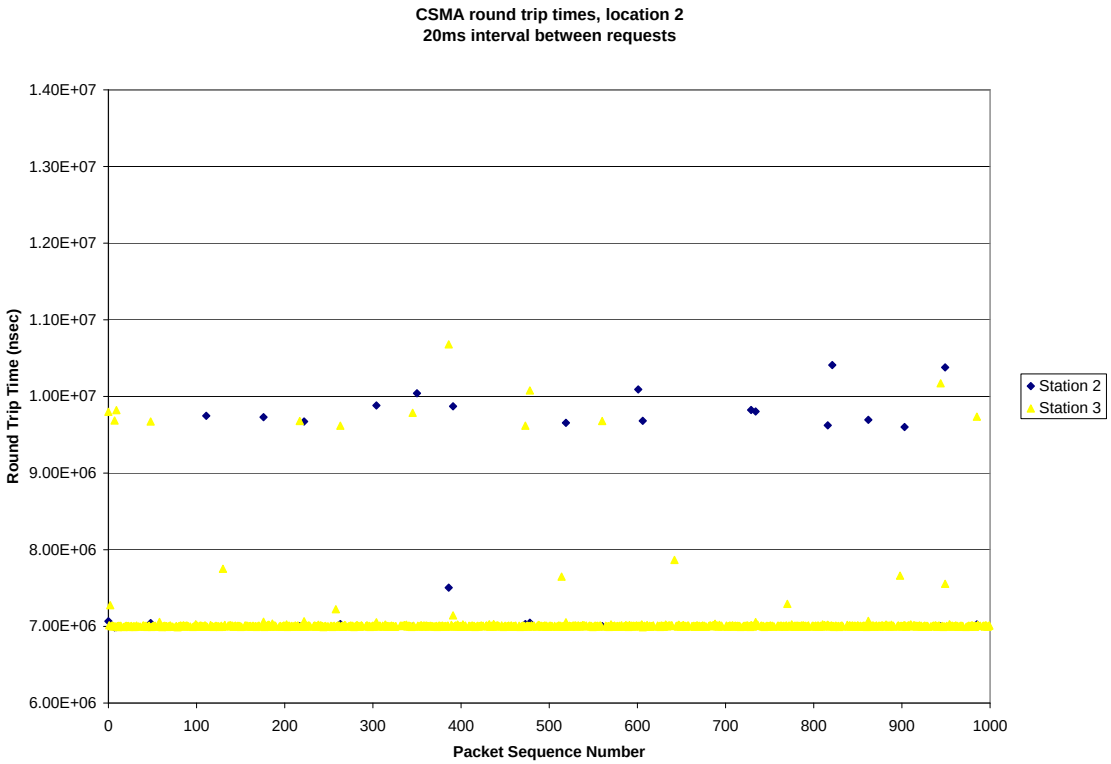

Fig. 6.5: Plot of heavy contention using CSMA at location 2

advantage. While these results clearly show that station 3 has some form of advantage, on occasion station 2 takes the leading role, as shown in fig. 6.6. There is no difference between the configuration of either station merely that the experiment was carried out on a different day.

It is also interesting to note that in the heavy interference environment the advantage station 3 has is noticeably reduced by approximately 10% owing to interference on the medium which gave the slower responding machine a chance when the faster machine (station 3) saw the medium as busy due to interference.

Location	Period	Station 2	Station 3	% loss overall	% bias in favour
Office	20ms	107/1000	948/1000	36%	Station 3 89.9%
Garden	20ms	29/1000	870/1000	29.9%	Station 3 96.8%
Office	10ms	156/1000	875/1000	34.4%	Station 3 84.9%
Garden	10ms	32/1000	847/1000	30.2%	Station 3 96.5%

Table 6.2: Analysis of CSMA contention

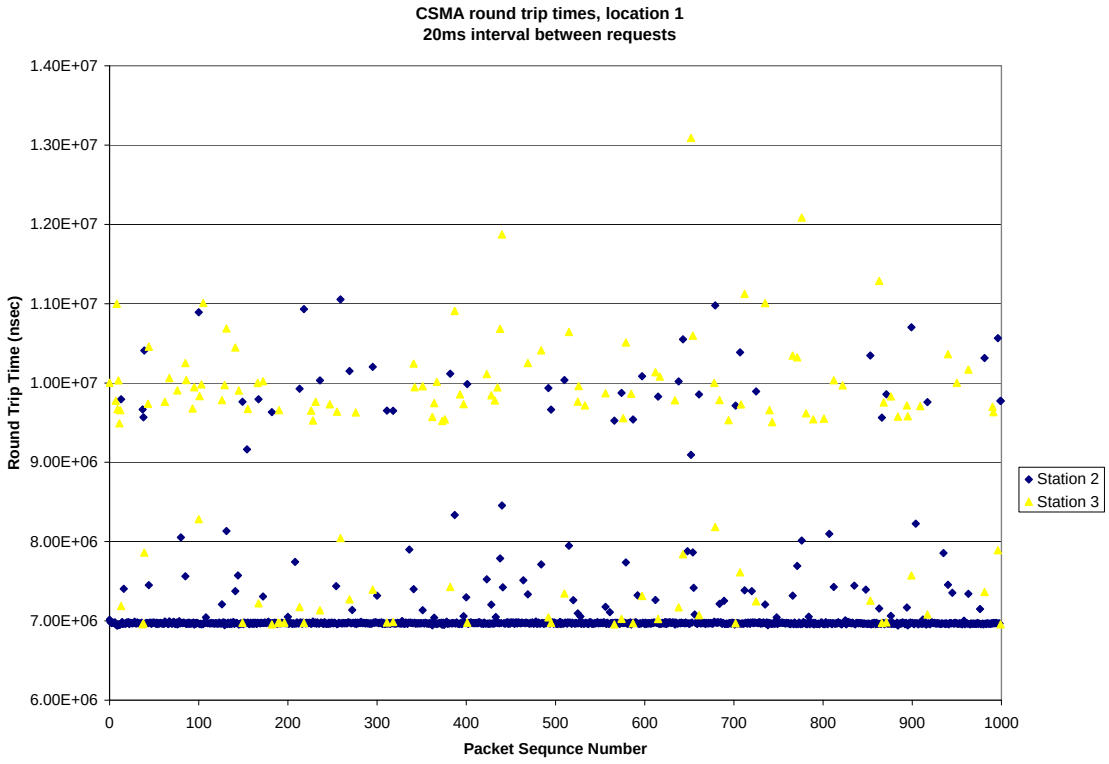

Fig. 6.6: Station 2 occasionally wins

6.5 TDMA experiments

TDMA experiments where conducted in a very similar fashion to those of the CSMA experiments discussed above. The experiments consisted of an 8 slot TDMA table, slot 0 was allocated to station 1 the master station, station 2 was allocated slot 1 and similarly station 3 was allocated slot 2. A slot time significantly more than the estimated propagation time was selected and was gradually reduced until performance started to degrade. As in the CSMA experiments detailed above, the master station sends one packet per cycle and the slaves respond to this request in there respective allocated slots.

In TDMA mode the packets are again sent as part of a periodic task, upon receipt the slave as in the CSMA case changes the originator identity. The packet is then queued, an independent periodic send task then sends the queued packet back to the master using the TDMA slot which has been assigned to it. In this scenario a station is designated as the leader or requestor, this station may or may not be the time synchronisation master. As shown in fig. 6.7, for a slot time of 3.5ms there is little or no contention.

As shown in fig. 6.8 as the slot time approaches the propagation time of 3.07ms performance begins to seriously degrade. In this case the degradation follows from a failure of the clock synchronisation protocol owing to excessive contention on the medium. The clock on station 2 clearly begins to diverge. By packet sequence number 900 the clock on station 2 has drifted so severely that its packets are still on the medium while station 3 tries to transmit leading to a high concentration of medium contention from this point onwards.

6.5.1 TDMA Round Trip Results

TDMA results where very much as expected, overall only one packet was lost during the experiments, fig. 6.7 clearly shows the slotted structure of the TDMA MAC. Each station responds in its allocated slot. Table 6.3 details the number of contention events in the TDMA environment over 1000 packet transmissions. Station 3 suffers more contention that station 2 since contention delays packets being sent by station 2, that delay leads to contention when

station 3 tries to transmit.

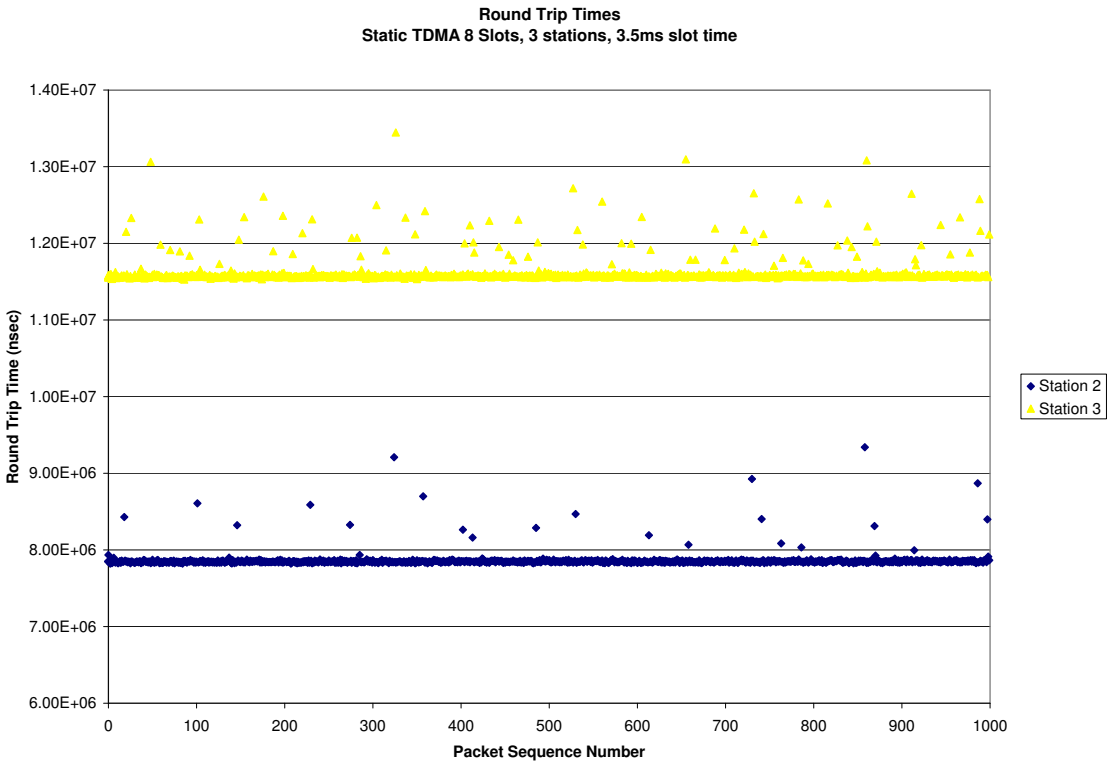

Fig. 6.7: TDMA in a steady state with a small amount of contention

The TDMA experiments highlighted an important practical concern. If a request is made in slot 1 to a station which has slot 2 allocated to it for example, it is not possible to guarantee the station would have sufficient time to compute a response thus the maximum response time is equal to, in the worst case of (CYCLE TIME + SLOT TIME) assuming that the processing delay to generate a response is less than one slot time, this bound will increase by one SLOT TIME for a corresponding increase in processing delay.

The TDMA performance is restricted heavily by the presence of the underlying 802.11 MAC layer. As is clear to see in experiments performed at location 1, the high interference location a substantial number of packets are delayed by interference (figs. 6.7 & 6.8). Since

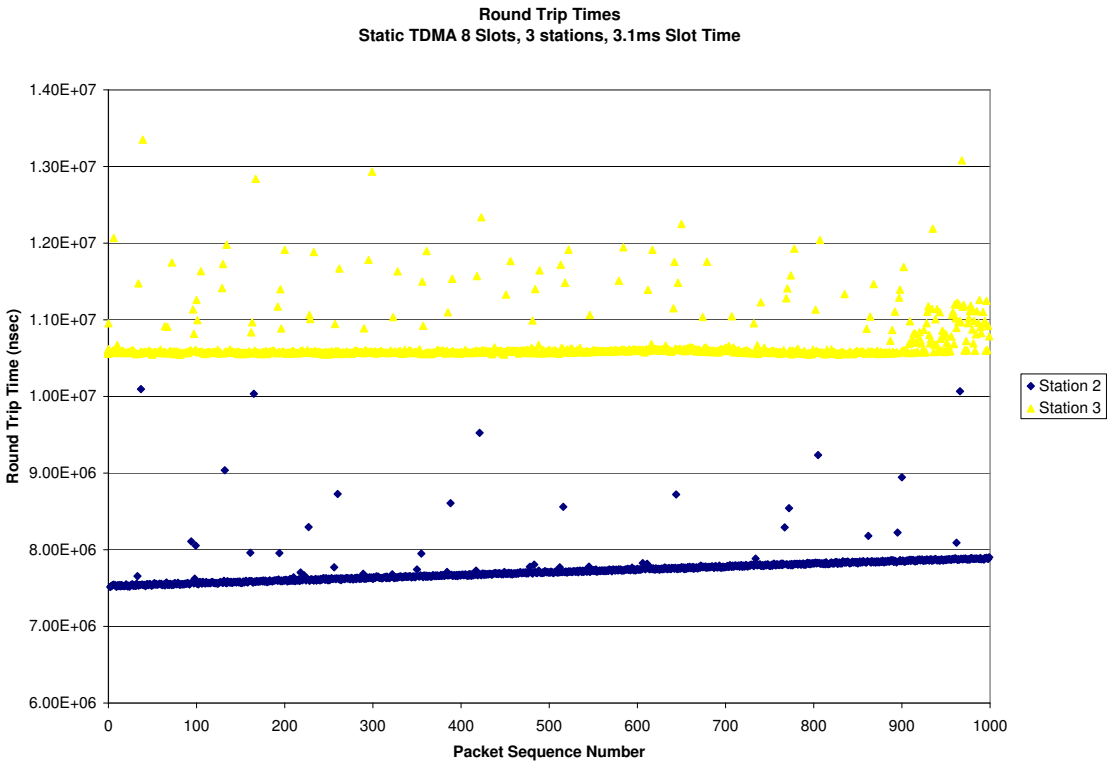

Fig. 6.8: Effect of excessive contention leads to clock drift

it is not possible to manipulate the operation of the CSMA MAC layer the level of back off calculated is beyond our control. Ideally in the presence of contention the assumption would be that it is localised interference and to transmit regardless. The distinction between a system failure and the result of interference can be seen in fig. 6.9 which was recorded at location 1, the interference free environment.

Fig. 6.9 shows a periodic event every 128 packets (3.072 msec) resulting in a delay. The underlying RTAI task executes at the appropriate time but the packet is delayed en route to the destination station. There could have been interference on the medium or more likely a processing delay copying the packet into the wireless card or a delay in the generation an processing of the packet received interrupt on the receiving station. Taking all these factors

Slot Time	Station 1	Station 2	Overall % Contention
3.5ms	23	89	11.2%
3.4ms	46	57	10.3%
3.35ms	57	75	13.2%
3.3ms	57	73	13.0%
3.25ms	70	286	35.6%
3.2ms	20	36	5.6%
3.15ms	82	93	17.5%

Table 6.3: Analysis of TDMA contention at location 1

into account the likely cause is periodic delay on one of the stations caused by a higher priority event having taking control, e.g. the hard disk buffers flushing the experiential data to disk will prevent access to the wireless card until the operation completes since it locks the main system bus of the computer while the operation is underway. There are a vast number of possible issues which could cause a delay,

- Delay on sender caused by another event on the system bus
- Interference on the medium
- Another packet on the medium
- Delay raising the packet received interrupt on the receiving station
- Delay on receiver caused by another event on the system bus

6.6 Comparision of CSMA & TDMA performance

Packet loss in CSMA was extremely high, up to 36% (table. 6.2) of all packets transmitted where lost. TDMA lost only 1 packet over the entire set of experiments leading to a exceptional level of reliability. A heavy bias was shown in favour of station 3 in the results presented here this was unexpected. Clearly bias in favour of a particular station is unac-

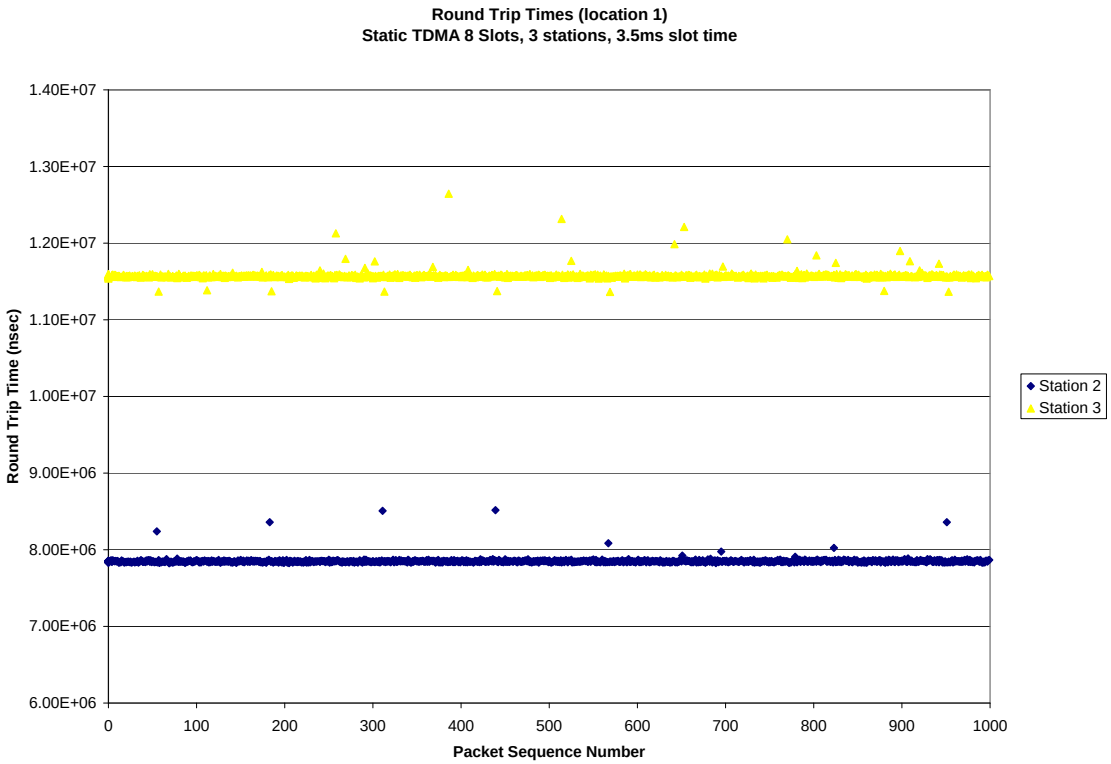

Fig. 6.9: In a interference free environment some packets are still delayed

ceptable in a network where each station is of equal importance. In evaluating the respective performance of CSMA and TDMA, it is important to remember that CSMA and TDMA have very different design goals.

TDMA showed a limited amount of contention, this was unexpected. While at first glance it may appear to be a failure of the clock synchronisation protocol the delays experienced where an order of magnitude greater than the clock synchronisation bound of $\pm 28\mu$sec. Since the 802.11 CSMA MAC layer is still present and active in the TDMA implementation, any interference on the medium will lead the CSMA MAC protocol generating a random back off counter. This leads to further contention in the next slot time. Under these conditions two options exist if the CSMA back off procedure can be disabled. Ignore the fact the medium

is seen as busy and assume that it is local interference and transmit the packet or secondly assume there actually is a packet currently on the medium and to drop this packet. Neither approach is available using the current wireless card driver.

In lightly loaded conditions CSMA will offer a much faster round trip time, however as the network load increases so will contention and packet loss will increase rapidly to an unacceptable level. In the TDMA environment packet loss in not an issue as shown above over 6 experiments in which a total of 18,000 packets where sent only a single packet was lost. In 4 CSMA experiments a total of 4,136 packets from a total of 12,000 where lost, an average loss of 34.5%.

Every packet sent under the CSMA protocol lead to contention for the slower station leading to a contention rate of 50%. TDMA offered a substantial improvement with contention generally less than 15% on average. 0% was expected but performance of the TDMA protocol was hampered due to the practical difficulties encountered in the implementation and in particular the retention of the underlying CSMA MAC protocol.

TDMA trades off response time in favour of reliability and deterministic behavour, a late packet is equally as bad if not worse than a lost packet. So while the response time to a request is a function of the current slot allocation state it is bounded and known in advance. In short in applications such as vehicle platoons[14] where timely delivery of messages is required and where high levels of message loss are unacceptable cannot be supported by a CSMA MAC layer. While on average TDMA has a slower response time owing to its slotted structure, it is bounded, the time is known in advance and importantly packet loss is effectively zero. Performance remains constant regardless of the presented load.

6.7 Timebounds in TBMAC

This set of experiments involved evaluating the time bounds of slot management functions. The TBMAC protocol[13] specifies that the time required to request a slot and to transmit in the subsequently allocated slot is no more than 5 * CYCLE PERIOD, where the CYCLE

PERIOD is equal to the sum of the CP and CFP if a CP is provided or the CFP alone if the CP is not implemented. In order to tolerate one failure of the atomic broadcast protocol the delay between the CFP slot the request is first transmitted in and its final delivery time is 2 * CYCLE TIME. To allow for the fact a request to allocate a slot may have to wait up to one CYCLE PERIOD before it can be sent in a CFP, a further CYCLE PERIOD will pass before the higher level that requested the slot allocation can actually transmit its request. Similarly for a slot deallocation the time bound is 3 * CYCLE PERIOD.

The time to allocate the slot was measured from the actual time the request to allocate a slot was received by the slot manager and the time the slot manager notified the level above it that the request has been successful or unsuccessful. This time was recorded for a series of 20 slot management requests. The worst delay occurs when a station has only one slot currently allocated to it and a request for an additional slot occurs a faction after the slot time of the currently allocated slot begins, this forces a wait of one full CYCLE PERIOD before the request can be broadcast.

This set of experiments concentrated on proving the time bounds detailed by the TBMAC protocol where practically achievable. While the time bounds are achievable the probability that they can be achieved is dependant on the configuration of the system and in particular the expected rate of slot management requests. This is the deciding factor in the number of atomic broadcast entries the packet header will contain. These factors are application specific and depend in part on the level of mobility a station is likely to experience. Based on practical experience and real world factors the upper time bound till a slot allocation should be 3 * CYCLE PERIOD + 1 CFP SLOT TIME, this is to allow for the fact it takes a finite amount of time to perform the slot allocation/deallocation once the request is delivered. The overall bound of 4 * CYCLE PERIOD remains valid.

6.7.1 Results

A total of 20 requests where made, each allocation request was followed by a deallocation request, all requests where successful and all where completed within 3 * CYCLE PERIOD.

All stations had a consistent view of slot allocations at the end of the experiments. These are simple experiments, the probabilities in achieving a slot allocation within the defined bounds is application specific and is heavily influenced by the expected rate of slot management requests which is linked to the degree of mobility stations enjoy which effects the expected rate of arrival and departure from a cell. Since this implementation supports only a static membership this element could not be evaluated This expected rate combined with the required probability in achieving the request begin carried out is heavily influenced by the number of atomic broadcast requests that can be carried in the TBMAC header.

6.8 Environmental Effects

It is very difficult to classify the effect of the environment. Packet loss was not an issue in either the office or garden location in the TDMA environment. Packet loss sees a significant increase as contention for the medium increases in both cases, this was expected. Interestingly packet loss reduces in CSMA as external interference increases. Due to the random nature of the back off timer and interaction between the stations it is not possible to duplicate any experiments.

There is a noticeable difference in the distribution of round trip times with the garden environment showing significantly fewer outliers. The office environment as described in 6.1 has many potential sources of interference, they are of course potential and it is very impossible to determine which are present at any one time, but clearly they have substantial negative effect on performance. Fig 6.7 & fig 6.9 clearly show the difference in terms of interference induced contention between the two locations.

6.9 Summary

Results are very similar to those expected, the results for CSMA where very extremely poor. TDMA results where impressive and show the clear benefits in terms of both stable delivery delays and virtually 0% packet loss. There is however a price to pay in terms of efficiency,

since the underlying MAC layer is still CSMA adjustments to various parameters such as the DIFS interval and contention windows are not possible, in a ideal situation a packet would be dropped instantly if the medium was found active at the scheduled transmission time. By allowing back off a domino like effect is created possibly leading to contention in the next slot and so on.

Chapter 7

Conculsions

An innovative approach has provided the ability to provide tight clock synchronisation in an ad hoc environment. With only a single modification to the source code of RTAI it was possible to provide the ability to synchronise the clocks of a number of stations to a single designated master. Through the deployment of this clock synchronisation protocol it was possible to deploy a TDMA MAC layer. Using this TDMA MAC layer a subset of the TBMAC protocol was implemented.

7.1 Summary

A core element of the TBMAC protocol the contention free period, the CFP has been implemented. Cristian's atomic broadcast protocol[10] has also been implemented thus allowing distributed agreement on slot management functions. The current implementation allows for a static membership with one station designated as a master during the bootstrapping phase. The other stations designated as slaves are required to preset their slot allocation tables such that the master station is allocated global TDMA slot 0. When started slaves allocate the master its slot and enter a listening phase. This listening phase serves a dual purpose, firstly to await reception of slot management messages transmitted using the atomic broadcast protocol and secondly to gain clock synchronisation to the clock of the master station. After

two full cycle periods the messages from the master are delivered on the slaves and those slaves which have been allocated slots during the bootstrapping phase may transmit. From this point onwards slot management becomes fully distributed and stations may request slot allocations and deallocations as determined by the application layer above the slot manager.

Membership is static however stations may choose to leave either by deallocating all there allocated slots through the `DEALLOCATE_SLOT` slot management message, or by explicitly by sending a `LEAVE` slot management message. Once a station has deallocated all its slots it cannot rejoin the cell since it does not have a slot to send a request in. All slot management requests are piggybacked on a data packet, if no data is available an empty packet known as the `NULL` packet will be transmitted. There is no loss of efficiency or channel capacity doing so since the slot is not available to any other station.

7.2 Contribution

This dissertation takes the up until now only simulated TBMAC protocol and implements it in the real world in real-time, thus allowing the TBMAC protocol to be evaluated under real world conditions. This is a significant departure from current and past work on MAC layers which has almost universally relied on the results of simulations alone in evaluation[7, 12, 26, 29, 35]. True TDMA in real time in an ad hoc environment with dynamic and distributed agreement on slot allocations has been implemented and evaluated.

TBMAC offers a contention free MAC protocol, message delivery is time bounded as are all slot management request. There are few if any collisions, the small number of collisions are attributable to environmental effects which are exacerbated due to the fact the TDMA MAC protocol is implemented above the standard 802.11 CSMA MAC protocol.

7.3 Completed Work

A clock synchronisation protocol has been devised, implemented and evaluated with very encouraging results. Master slave based TDMA was then deployed. Based on the master slave

implementation of slot allocation a subset of the TBMAC protocol has been implemented and evaluated. TBMAC's performance in the real world reinforces the results of the evaluation of the NS2 implementation presented in [13].

7.4 Future Work

Only a subset of the TBMAC protocol has been presented here, the next step would be to implement the contention period, the CP thus allowing fully dynamic membership, once the contention period and contention free period have been implemented inter cell communication could be implemented, as is discussed below implementation of intercell communication has significant technical challenges.

The current clock synchronisation protocol is simplistic yet effective it however could be significantly improved by including some of the techniques used by NTP which smooth out oscillations in the updates to the clock.

The primary restriction in performance is the fact that the TDMA structure is overlaid over the CSMA/CA scheme of 802.11, in particular the inability to control or manipulate the internal timing used by the 802.11 MAC, a frequent approach used in some recent work[29, 35, 26] is not available to us.

It was not possible to disable the CSMA exponential backoff procedure used in face of contention this lead to a domino effect if one packet encountered what it thought to be contention it would back off for up to 625μsec this would lead to a further contention since the packet would still be on the medium during the next slot time. This combined with the fact the MAC layer is implemented in software there are significant latencies and inefficiencies. The act of transferring an outgoing packet to the wireless cards buffer memory is quite a significant event during which the card can do no other work.

These problems could be overcome if a way was found to implement a portion is not all of the MAC protocol in the hardware of the wireless card. Such an approach would allow the critical elements of the TBMAC protocol such as the TDMA slot table to be implemented

in hardware thus virtually eliminating the majority of the latencies and non deterministic behavour that currently exists during interaction between the wireless card and the host computer.

A major recurring issue through this work is that of clock synchronisation and in particular the practical difficulties in doing so in a TDMA environment as outlined in chapter 5.3. Inter cell communication is not dealt with in the current implementation due in part to clock synchronisation but also due to major difficulties at the physical level of the wireless card. To act as a bridge between cells a station needs to be effectively on two networks at once. This requires rapid alteration of the frequency channel the card uses, the current real time driver for the card does not have an API to directly alter or query the frequency channel setting. In addition 802.11 devices cards are not specifically designed to rapidly change frequencies. The effect of changing frequency and in particular the latency in doing so is currently unknown.

Bibliography

[1] HiperLAN2 Global Forum. http://www.hiperlan2.com/.

[2] NTP: The Network Time Protocol. http://www.ntp.org/.

[3] RTAI: Real Time Application Interface. http://www.aero.polimi.it/∼rtai/.

[4] RTnet:Hard Real-Time Networking for Linux/RTAI. http://www.rts.uni-hannover.de/rtnet/.

[5] The Network Simulator NS-2. http://www.isi.edu/nsnam/ns/.

[6] Andrew S Tanenbaum. *Computer Networks*, 4th ed. Prentice Hall, 2002.

[7] Bharghavan, V., Demers, A., Shenker, S., and Zhang, L. MACAW: a media access protocol for wireless LAN's. *SIGCOMM Comput. Commun. Rev. 24*, 4 (1994), 212–225.

[8] Carley, T. W., Ba, M. A., Barua, R., and Stewart, D. B. Contention-Free Periodic Message Scheduler Medium Access Control in Wireless Sensor / Actuator Networks. *IEEE Real-Time Systems Symposium* (Dec 2003), 298–307.

[9] Cristian, F. Probabilistic Clock Synchronization. *Distributed Computing 3*, 3 (1989), 146–158.

[10] Cristian, F. Synchronous Atomic Broadcast for Redundant Broadcast Channels. *Journal of Real-Time Systems 2* (1990), 195–212.

[11] Cristian, F., and Fetzer, C. Integrating external and internal clock synchronization. *Real-Time Systems*, 2 (1997), 123–171.

[12] Crow, B., Widjaja, I., Kim, L., and Sakai, P. IEEE 802.11 Wireless Local Area Networks. *IEEE Communications Magazine 35*, 9 (Sep 1997), 116–126.

[13] Cunningham, R. *Time Bounded Media Access Control for Ad-Hoc Networks*. PhD thesis, University of Dublin, Trinity College, Oct 2003.

[14] Cunningham, R., and Cahill, V. System support for smart cars: requirements and research directions. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop* (2000), ACM Press, pp. 159–164.

[15] Cunningham, R., and Cahill, V. Time bounded medium access control for ad hoc networks. In *Proceedings of the second ACM international workshop on Principles of Mobile Computing* (2002), ACM Press, pp. 1–8.

[16] Cunningham, R., and Cahill, V. Time bounded medium access control for ad hoc networks,, 2002. http://www.dsg.cs.tcd.ie/uploads/catergory224/39.ppt.

[17] Doufexi, A., Armour, S., Karlsson, P., Nix, A., and Bull, D. A Comparision of HIPERLAND/2 and IEEE 802.11a. Centre for Commmunications Research, University of Bristol, UK.

[18] Elson, J., Girod, L., and Estrin, D. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev. 36*, SI (2002), 147–163.

[19] Gusella, R., and Zatti, S. The berkeley unix 4.3bsd time synchronization protocol. Tech. rep., University of California at Berkeley, 1985.

[20] Hanssen, F., van den Boom, J., Jansen, P. G., and Scholten, H. Time synchronization for an ethernet-based real time token network. In *Proceedings of the second International Workshop on Real Time LANs in the Internet Age (ECRTS/RTLIA03)*

(Jul 2003), The European Association for Microprocessing and Microprogramming, Polytechnic Institute of Porto, pp. 71–74.

[21] Hughes, B., and Cahill, V. Towards Real-time Event-based Communication in Mobile Ad Hoc Wireless Networks. In *Proceedings of 2nd International Workshop on Real-Time LANS in the Internet Age 2003 (ECRTS/RTLIA03)* (Jul 2003), Polytechnic Institute of Porto, pp. 77–80.

[22] IEEE. *IEEE std 802.11 Wireless LAN medium access control (MAC) and Physical Layer (PHY) Specification*. IEEE Computer Society, 1997.

[23] IEEE. *IEEE Std. 802 Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band*. IEEE Computer Society, 1999.

[24] IEEE. *IEEE Std. 802 Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*. IEEE Computer Society, 1999.

[25] INTEL. *Intel Architecture Software Developers Manual Volume 2, Instruction Set Reference*, 1997. http://developer.intel.com/design/pentium/manuals/24319101.pdf.

[26] Joao L Sobrinho and A. S. Krishnakumar. Real-Time Traffic over the IEEE 802.11 Medium Access Control Layer, 1996.

[27] Johnsson, M. HiperLAN/2 The Broadband Radio Transmission Technology Operating in the 5 GHz Frequency Band, 1999.

[28] Karn, P. MACA - A New Channel Access Method for Packet Radio. In *In proceedings of the 9th ARRL Computer Networking Conference* (Sep 1990), pp. 134–140.

[29] Lin, C. R., and Gerla, M. Real-time support in multihop wireless networks. *Wireless Networks 5*, 2 (1999), 125–135.

[30] Mills, D. The network computer as precision timekeeper. In *Proceedings of Precision Time and Time Interval (PTTI) Applications and Planning Meeting* (Dec 1996), pp. 96–108.

[31] Mullender, S. J., Ed. *Distributed Systems*, 2nd ed. Addison-Wesley, 1993.

[32] O'Connor, N. Clock Synchronization for Multihop Wireless Networks. Master's thesis, University of Dublin, Trinity College, Sep 2003.

[33] Rahnema, M. Overview of the GSM system and protocol architecture. *IEEE Communications Magazine 31*, 4 (Apr 1993), 92–100.

[34] Rakity, P., Taylor, L., and Lynn, K. A Distributed Time Bounded Service, Jan 1994. Submission IEEE P802.11-94/21 in respect of Wireless LAN Medium Access Control and Physical Layer Specifications.

[35] Rusty O. Baldwin and Nathaniel J. Davis IV and Scott F. Midkiff. A real-time medium access control protocol for ad hoc wireless local area networks. *SIGMOBILE Mob. Comput. Commun. Rev. 3*, 2 (1999), 20–27.

[36] T Konefal AND C A Marshman and L M McCormack. Potential Electromagnetic Interfence to Radio Services From Railways Final Report AY4110. Tech. rep., University of York, May 2002.

[37] Zou, S., Wu, H., and Cheng, S. A New Mechanism of Transmitting MPEG-4 Video in IEEE 802.11 Wireless LAN with DCF, Apr 2003.