EXPLICITLY PIPELINING IEEE 802.11 TO ENHANCE PERFORMANCE

BY

PRIYA RAVICHANDRAN

B.S., University of Illinois at Urbana-Champaign, 2002

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

Explicitly Pipelining IEEE 802.11 to Enhance Performance

Approved by
Dr. N. H. Vaidya

_____

_____

ABSTRACT

Channel capacity is a scarce resource in wireless networks, and the Medium Access Control (MAC) protocol directly impacts the utilization of this resource. Unfortunately, the random backoff algorithm and RTS/CTS control packets used by IEEE 802.11 DCF result in a large overhead that translates into low channel utilization. Furthermore, when the network is highly loaded, the bandwidth wasted due to collisions is also significant. Prior work has proposed various methods to reduce this bandwidth wastage – from modifying the backoff algorithm to employing a multistage contention-resolution algorithm to reduce wastage due to collisions. In this thesis, we evaluate the Dual Channel MAC (*DCMAC*) protocol that uses an explicit pipelining structure to increase utilization. This is achieved by resolving the contention over one channel while simultaneously transmitting the data packet over another channel. We identify the various problems that result from the use of two channels and discuss possible solutions to these issues. We use simulations to show how this protocol performs under various network topologies. Results show that this dual channel scheme performs better than IEEE 802.11 in wireless LANs, though the performance improvement is dependent on the bandwidth division, packet size distribution, and the interdependence of the flows.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Nitin Vaidya, and Ms. Xue Yang for their invaluable advice and guidance in preparing this thesis. I would also like to thank my family and friends for their suggestions, support, and unwavering confidence in me.

TABLE OF CONTENTS

LIST OF TABLES vi

LIST OF FIGURES vii

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 OVERVIEW OF IEEE 802.11 DCF 5

CHAPTER 3 RELATED WORK 8

CHAPTER 4 DCMAC PROTOCOL DESCRIPTION 17
- 4.1 Overview 17
- 4.2 Definition of Terms 18
- 4.3 RTS/CTS Packet Modification 19
- 4.4 *DCMAC* Details 20
- 4.5 Discussion of Issues Arising from *DCMAC* 25
  - 4.5.1 Deafness 25
  - 4.5.2 Differing channel conditions 27

CHAPTER 5 PERFORMANCE EVALUATION 30

CHAPTER 6 CONCLUSION AND FUTURE WORK 40

REFERENCES 43

LIST OF TABLES

Table **Page**

5.1 DSSS specifications 30

LIST OF FIGURES

Figure		Page
2.1	A single dialog, using the RTS/CTS access method.	6
3.1	Example of three-stage pipelining.	14
4.1	Overview of pipelining.	18
4.2	Illustration of terms.	19
4.3	An example with *DCMAC*.	22
4.4	An example of how using available information regarding the current sender and receiver reduces channel capture. A value of 1 on the *Channel Access* axis is an indicator that the flow currently has access rights to the data channel.	24
4.5	Deafness in the common receiver case.	26
5.1	Example of the three different scenarios simulated.	31
5.2	Aggregate throughput versus control bandwidth for disjoint flows in a WLAN.	32
5.3	Aggregate throughput versus control bandwidth for the access-point topology in a WLAN.	36
5.4	Aggregate throughput versus control bandwidth for the random topology in a WLAN.	37

CHAPTER 1

INTRODUCTION

Wireless networks are fast becoming popular as they allow users the ability to remain connected while on the move. These wireless networks can either be infrastructure-based or ad hoc networks. Infrastructure-based networks involve wireless devices communicating with an access point (AP), which then connects to the Internet via a wired network. On the other hand, ad hoc networks are a collection of devices outfitted with wireless transceivers that enable communication between the nodes,[1] independent of any pre-existing infrastructure. This independence is a major benefit as it enables communication in situations where there is no time to set up the necessary infrastructure, e.g., military and rescue operations, or in situations where the need for a communication network is temporary, e.g., conferences.

Wireless networks face a multitude of challenges that the traditional wired networks do not. Wired networks typically communicate over wires made from copper or fiber optics, where the probability of error is very small. Collision detection is generally fast since nodes are able to listen to the medium while they are transmitting. The topology of a wired network is usually fixed, which greatly simplifies routing. Furthermore, wired networks have dedicated routers whose sole function is to ensure that the packets are correctly routed from source to destination. In contrast, wireless networks communicate over a highly error prone medium. Collisions become harder to detect since the wireless

[1] Henceforth these devices shall be referred to as nodes.

transceivers cannot receive reliably while transmitting on the same channel. As such, the absence of a reply is usually the only indication that a collision has occurred and the time required for detection is dependent on the length of the transmitted packet. The topology of an ad hoc network is constantly changing and node membership within the network is also not fixed. This makes neighbor discovery and routing nontrivial. Furthermore, network connectivity is dependent on nodes voluntarily forwarding packets for each other, in order to route a packet from source to destination. These are just some of the many challenges that wireless networks face. In this thesis, we focus on the issue of medium access control in wireless networks.

The function of medium access control (MAC) is to enable the effective sharing of the channel among all nodes in the network. Ideally, for both wired and wireless networks, the MAC protocol should facilitate perfect channel utilization while still being fair to all flows within the network. The conflict between these goals usually does not allow the ideal to be realized in practice.

In wireless networks, MAC protocols can be classified either as reservation-based protocols or contention-based protocols. Reservation-based protocols are used in situations where an access point is present. The AP polls the nodes before assigning access rights in turn and a node is only allowed to transmit when it is allocated the right to do so. These protocols are also used to provide quality of service (QoS) guarantees for real-time traffic. An example of a reservation-based protocol is the IEEE 802.11 Point Coordination Function (PCF). Contention-based protocols are distributed algorithms that allow nodes in an ad hoc network to be able to communicate effectively over the shared channel and are largely based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). ALOHA, HIPERLAN, MACAW, and IEEE 802.11 Distributed Coordinated Function (DCF) are just some of the many contention-based protocols that have been proposed. In general, contention-based protocols are preferred as channel capacity is wasted if the flow that currently is allocated transmission rights under the reservation-based protocol

underutilizes the channel. Furthermore, contention-based protocols will be better able to adapt to meet the dynamic demands of a wireless network. Accordingly, the European Telecommunication Standards Institute (ETSI) has adopted HIPERLAN as the standard for wireless LANs (WLANs), while IEEE 802.11 has been adopted as the international standard for WLANs.

New MAC protocols are constantly being developed in an attempt to approximate the ideal as closely as possible. Some protocols optimize the system throughput at the expense of fairness while others give preference to achieving a greater level of fairness in wireless networks. Given that protocols are continually being proposed, various metrics have also been developed in order to be able to evaluate the efficiency and robustness of these protocols. One common metric is channel utilization, which is defined to be the fraction of time spent on successful transmissions. Another metric is saturation throughput, which represents the maximum throughput the system can achieve under stable conditions. Both channel utilization and saturation throughput are directly affected by the amount of overhead a protocol introduces for each successful transmission and the efficiency with which collisions are resolved. Protocol fairness is also an important issue that considers whether a protocol allows all flows an equal chance of accessing the channel and is often considered to be a critical characteristic of the protocol. IEEE 802.11 is known to exhibit short-term unfairness but evens out in the long term. Some common fairness measures include the Max/Min ratio and Jain's Fairness Index [1].

In this thesis, we evaluate a new protocol that attempts to improve upon the performance of IEEE 802.11 while still maintaining the level of fairness that IEEE 802.11 does. The protocol is only considered for WLANs, where all hosts are within one hop of each other. The rest of this thesis is organized as follows. In Chapter 2 we present an overview of the IEEE 802.11 DCF function. The discussion of some of the related work on MAC protocols is presented in Chapter 3. In Chapter 4, we propose a dual channel MAC protocol, *DCMAC*, that exploits the principle of pipelining, and we discuss the various

issues that arise as a result of adopting this pipelined structure. Simulations are used to evaluate the performance of *DCMAC* in WLANs and the results are presented in Chapter 5. Chapter 6 concludes this thesis with a brief summary and a discussion of possible directions for future work.

CHAPTER 2

OVERVIEW OF IEEE 802.11 DCF

IEEE 802.11 Distributed Coordinate Function (DCF) is a random access protocol based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). The standard [2] defines two access methods — the basic access method and the RTS/CTS access method. The basic access method involves only the exchange of the Data packet and the ACK packet. The RTS/CTS access method extends the basic access method with the exchange of a Request-To-Send (RTS) control packet and a Clear-To-Send (CTS) control packet before the actual exchange of the Data and the ACK. This extension facilitates faster collision detection, especially when the data packets are large. It also solves the hidden terminal problem[1] experienced in wireless networks. Here we present the main features of the IEEE 802.11 DCF, with respect to the RTS/CTS access method. For further details, please refer to [2].

The IEEE 802.11 DCF uses the binary exponential backoff (BEB) algorithm to resolve channel contention. A station that wants to transmit a data packet must first carrier-sense the medium. If the channel is idle for at least a DCF Interframe Space[2] (DIFS) duration, the node randomly picks a backoff counter value. This value is drawn from a uniform distribution over the interval [0, CW-1], where CW is the size of the node's contention window. If the medium is sensed to be busy, the node must defer its transmission

[1] Refer to [3] for a discussion on the hidden terminal problem.
[2] The various interframe spacings are defined in the standard.

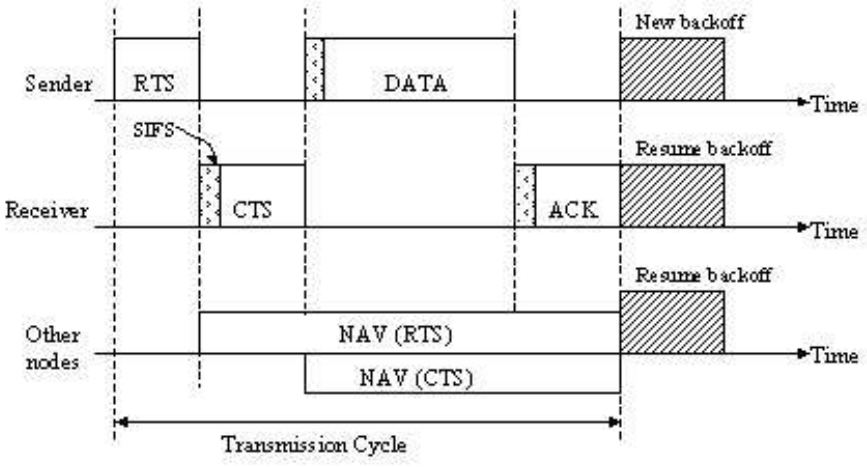


Figure 2.1 A single dialog, using the RTS/CTS access method.

until the medium is idle for at least a DIFS duration before selecting a backoff value.

The backoff value represents the number of idle slots[3] a node has to wait before it can transmit. For each idle slot it senses on the channel, the node decrements its backoff counter by one. If the channel is sensed to be busy, the node freezes its backoff counter until the channel is idle for at least the DIFS duration before it resumes decrementing the counter.

When the backoff counter reaches 0, it transmits an RTS. If a node receives an RTS, it replies with a CTS. Upon receiving the CTS, the node then transmits the Data packet, which is then replied with an ACK. This represents a single dialog, and a short Inter-frame Space (SIFS) is used to separate the various transmissions within a single dialog, as illustrated in Figure 2.1.

Both the RTS and the CTS packets contain the total duration of the dialog. All nodes that overhear either the RTS or the CTS set their Network Allocation Vector (NAV) for this duration and defer transmitting while their NAV is set. As a consequence, this

[3] A slot is a fixed duration of time defined in IEEE 802.11 and is PHY dependent.

allows for the collision-free transmission of the Data and ACK. This mechanism of deferring transmission based on the NAV is known as *virtual carrier sensing* and it effectively reserves the channel for the current dialog.

The combination of the virtual and physical carrier sensing mechanisms ensures that collisions only occur when two nodes pick the same slot to transmit. Since a node cannot sense the medium while transmitting, an absence of a reply (i.e., a CTS or an ACK) is taken to be an indication that a collision has occurred. When this happens, the node doubles its contention window and reselects a new backoff value. Nodes that sense the collision will set their NAV to the Extended Interframe space (EIFS) duration. This duration is defined to allow an ACK to be successfully transmitted before other transmissions recommence. As the number of collisions increases, nodes exponentially increase their contention windows so as to reduce the probability of another collision. A node starts with its CW=CW_{min} and with each collision, it doubles its backoff until it reaches the maximum value of CW_{max}. The values of CW_{min} and CW_{max} are dependent on the physical layer used. Once the node is successful in its transmission attempt, it resets its CW back to the minimum value. The total duration for one successful packet transmission, which is the interval between the end of the last successful transmission and the reception of the ACK for the current packet, is known as one transmission cycle.

Upon completing a successful transmission, the standard requires the node to wait a random backoff time between two packet transmissions. This is to prevent channel capture by a single node and to provide all nodes with a fair chance of accessing the channel in turn.

CHAPTER 3

RELATED WORK

This chapter presents some of the research that has been done in the field of medium access control (MAC) for ad hoc networks. Much of the research here is motivated by the fact that the prior research done with respect to wired networks is not directly applicable to the wireless arena. This is a direct consequence of the different properties of the wired and wireless medium.

One of the earliest MAC protocols proposed was ALOHA [3]. In pure ALOHA, nodes transmit whenever they have packets to send, without regard to the current state of the medium. A positive acknowledgment scheme is used to determine if the transmission was successful. This is a common feature in most MAC protocols designed for the wireless medium since the wireless transceiver does not allow a node to transmit and listen to the channel simultaneously. If no ACK returns, a collision is assumed to have occurred and the node retransmits after a random delay. Since no form of carrier sensing is done, a transmitted packet is vulnerable to collision for an interval equal to twice its transmission time.[1] Put differently, if a second transmission starts within a packet's vulnerable period, a collision occurs and both transmissions are unsuccessful. A throughput analysis shows that pure ALOHA utilizes at most 18% of the available channel bandwidth. A simple extension to pure ALOHA was to divide time into discrete intervals, known as slots, and allow nodes to only transmit at the beginning of a slot. This extension, known as slotted

[1] This duration is known as a packet's *vulnerable period.*

ALOHA, halves the vulnerable period of a packet and is able to achieve a maximum channel utilization of 36%, i.e., double that of pure ALOHA.

To further increase the channel utilization, carrier sense multiple access (CSMA) protocols were developed [3]. CSMA protocols can be divided into nonpersistent CSMA and p-persistent CSMA. In both categories, nodes first sense the channel before any action is taken. In nonpersistent CSMA, the node transmits if the medium is idle. If the medium is sensed as busy, the node waits a random delay period before retrying. In p-persistent CSMA, the node continually senses the medium until it is idle. It then transmits in a given slot with probability p and defers transmission to the next slot with probability $1 - p$. If a collision occurs, the node waits a random delay before retransmitting. In the limit, 1-persistent CSMA employs the greedy principle and always transmits as soon as it senses the channel as idle.

In [4], Kleinrock and Tobagi show that the CSMA protocols are superior to both pure and slotted ALOHA. CSMA protocols not only achieve significantly higher throughputs, they also perform better under high network loads. CSMA is also more efficient than ALOHA as it has smaller delay times[2] and requires a smaller number of retransmissions per successful transmission. Given these obvious benefits of incorporating carrier sensing into the MAC protocols, most of the succeeding protocols have been based on the CSMA model.

Karn [5] proposed multiple access with collision avoidance (MACA). Here, the RTS/CTS exchange precedes the actual data transmission. Nodes overhearing the RTS and CTS will defer their transmissions long enough to permit the successful reception of the CTS and the data packet, respectively. By doing so, both the exposed and hidden terminal problems are solved. If a node does not receive a CTS in reply to its RTS, it assumes that a collision has occurred and uses the binary exponential backoff algorithm (BEB) to

[2]Delay time is defined as the interval between packet generation and successful reception.

reschedule its transmission. MACA does not incorporate any form of acknowledgment scheme for the data packet. As such, the node transmitting the data packet will assume that the transmission was successful. Should the packet have been corrupted during transmission, recovery is the responsibility of the higher layers. This effectively increases the end-to-end delay experienced by the application.

The IEEE 802.11 DCF protocol [2] described in Chapter 2 is similar to MACA, except that it includes link-level acknowledgments to enable faster packet loss detection. If the ACK is not received, it is assumed that a collision occurred and IEEE 802.11 DCF performs the retransmission at the MAC level. This reduces the end-to-end latency experienced by the application.

Bianchi [6] investigates IEEE 802.11 DCF by analytically modeling the protocol, using a bidimensional Markov chain to capture the behavior of the BEB algorithm. While the model assumes finite nodes and ideal channel conditions, it is flexible enough to be able to model the basic access method, the RTS/CTS access method, or a hybrid of the two. The paper shows that IEEE 802.11 exhibits some instability. But, unlike the ALOHA protocols, the throughput asymptotically[3] approaches 68% of the maximum value as the offered load increases. Bianchi [6] also illustrates that the basic access method is strongly dependent on the number of nodes in the network and size of CW_{min}. On the other hand, the RTS/CTS access method is insensitive to variations in network size and contention window size. Furthermore, [6] proves that the RTS/CTS access method increases the efficiency of the protocol by significantly reducing the bandwidth wastage due to collisions.

The binary backoff algorithm employed in MACA and IEEE 802.11 has been shown to cause suboptimal system performance for a variety of reasons as discussed in [7] - [9].The BEB displays large variations in the size of the node's contention window, as it doubles the CW with each collision and then resets it back to the minimum value upon complet-

[3]This asymptotic value is the saturation throughput of the protocol.

ing a successful transmission. Furthermore, by resetting the CW back to CW_{min}, it does not attempt to perform any sort of flow control to reduce the probability of collisions in future transmission.

Bhargahavan et al. [8] proposed MACAW, based upon modifications to the MACA protocol. While similar to IEEE 802.11 DCF, MACAW adopts a modified version of the BEB so as to reduce the variations seen on the size of a node's CW. The new backoff algorithm is known as multiplicative increase linear decrease (MILD). For each collision, MILD increases a node's CW size by a multiplicative factor of 1.5 and linearly decrements the CW size after each successful transmission. This allows the size of the CW to increase rapidly, thus reducing the probability of collisions in a congested network, but the linear decrement reduces the large variations that BEB produces. By slowly decrementing the contention window size, MILD also attempts to include some sort of flow control in MACAW. Other features, such as contention window copying to propagate congestion information and the use of a "Data Sending" packet to replace the need for carrier sensing, were incorporated into MACAW to further improve performance.

In a more radical approach, [7] proposes *Fast Collision Resolution* (FCR). The motivation behind FCR is the perfect scheduling algorithm, where a node transmits with probability 1 if it is scheduled to do so and with probability 0 otherwise. When an active node[4] detects a collision, it doubles its CW and repicks its backoff value, regardless of whether it was involved in the collision. This reduces the probability of successive collisions, since the large disparity in CW sizes makes it unlikely that two nodes will transmit simultaneously. It also uses a smaller CW_{min} and larger CW_{max} than the values in [2]. The smaller CW_{min} allows the successful node to transmit numerous packets while reducing the idle time between transmissions. This effectively promotes channel capture by one flow. If the number of idle slots detected is greater than some threshold value, then the backoff counters are exponentially reduced, again to minimize the idle slots between

[4] Active nodes are those with packets to transmit.

transmissions. Thus, FCR shows a marked improvement over IEEE 802.11 DCF in terms of system throughput, achieved at the expense of fairness. But the authors show that, by incorporating the self-clocked fair queuing algorithm proposed by Golestani, FCR is able to maintain its performance gain while improving protocol fairness.

Cali et al. [9] analytically explore the dependence between the average contention window size and the maximum achievable throughput. The authors approximate the BEB algorithm with a p-persistent protocol. They also derive an algorithm that estimates the average contention window size. This estimate is used to obtain the value of p that maximizes the attainable throughput. Using this analysis, the authors show that IEEE 802.11 DCF operates far from its optimal capacity, as a result of the static set of sizes that the contention window assumes in the BEB algorithm. Furthermore, the paper also shows that the contention window size can be dynamically tuned to improve the capacity of IEEE 802.11 DCF, just by observing the network status[5] and exploiting the estimation algorithm derived here. Cali et al. [9] operate under the strong assumption that nodes are able to obtain perfect feedback about the network and channel conditions. Cali et al. [10] relax this assumption and proposes the *Dynamic IEEE 802.11* protocol. This protocol uses estimates of the network status to dynamically tune the size of the contention window. The authors show that, under steady state conditions, *Dynamic IEEE 802.11* operates very close to the theoretical capacity derived in [9]. Furthermore, they prove that the proposed protocol is robust enough to be able to rapidly recover from estimation errors and sudden changes in network size to converge back to its optimal capacity.

HIPERLAN [11], which is the ETSI standard for WLANs, uses blackbursts instead of a backoff algorithm to do contention-resolution. Based on CSMA/CA, HIPERLAN uses a channel access protocol known as Elimination Yield - Non-preemptive Priority Multiple Access (EY-NPMA) that divides the contention resolution into two stages, namely prioritization and contention. Since the protocol supports different priority levels, the

[5]This refers to whether the channel is sensed idle or busy.

prioritization stage ensures only nodes with the same priority levels contend with each other. The contention stage is further subdivided into an elimination stage and a yield stage. During elimination, the contending nodes emit a burst of random length, and only the nodes that emit the longest burst survive this stage. In the event that more than one node survives the elimination stage, nodes reduce collision probability by delaying their transmissions by a small random delay. If the channel remains idle at the end of the delay, the node transmits. Otherwise, the node loses the contention and retries the next round. Analysis of the HIPERLAN protocol [12] shows that the performance of EY-NPMA gracefully degrades as the network size increases. Furthermore, HIPERLAN remains stable under all types of traffic and the probability of a successful transmission is almost independent of network size.

Weinmiller et al. [13] conducted a performance study on both HIPERLAN and IEEE 802.11 DCF. They show that the protocols suffer throughput degradation and an increase in mean delay as the network size increases. While HIPERLAN has attempted to cater for priorities, it is not suitable for providing QoS guarantees as it is still a best effort protocol. IEEE 802.11 needs to have its PCF enabled before it can provide time bounded traffic, but HIPERLAN incorporates this into its normal operation via packet lifetimes. Unlike the RTS/CTS access method in IEEE 802.11 DCF, HIPERLAN does not have a way to overcome the hidden terminal problem.

In this thesis we attempt to improve the performance achieved by masking the contention overhead via pipelining. Pipelining allows the execution of various jobs to overlap in time. This is accomplished by dividing each job into stages, and as soon as the first job finishes a stage, the second job enters that stage as the first job moves on to the next stage. Figure 3.1 illustrates this concept by using a three-stage pipeline. In MAC protocols, the 'job' is the dialog to transmit a packet and it is divided into a *contention-resolution* stage and a *packet transmission* stage.

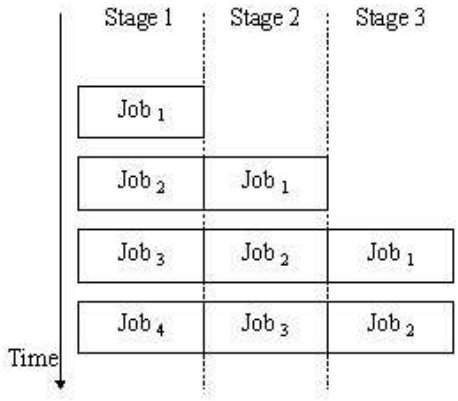


Figure 3.1 Example of three-stage pipelining.

While not widespread, the concept of pipelining in MAC protocols is not a new one. Todd and Mark [14] consider this issue for a generic MAC protocol. They divide each transmission cycle into a bandwidth-independent component, a bandwidth-dependent component, and the successful transmission. The bandwidth-independent component accounts for the idle slots within a transmission cycle while the bandwidth-dependent component accounts for the collisions that occur. In terms of these components, [14] analytically compares the capacity of the generic unpipelined MAC protocol with that of the generic pipelined MAC protocol. Their analysis shows that the pipelined protocol cannot do any better than the unpipelined protocol if there is no bandwidth-independent component in the transmission cycle. On the other hand, if there are no bandwidth-dependent components in the transmission cycle, pipelining does lead to a gain in performance. The performance gain decreases as the ratio of the idle overhead to the data packet size increases. Thus, [14] shows that that the benefit of pipelining will depend on how much of the transmission cycle is bandwidth-dependent and how much is bandwidth-independent.

Yang and Vaidya [15] propose Pipelined Packet Scheduling (PPS) that partially pipelines the contention resolution process over a data channel and a narrow busy tone channel. The contention resolution process is divided into two stages and both stages adopt the backoff algorithm, each with its own set of CW_{min} and CW_{max} values. All backlogged

stations start in the first stage and decrement the backoff counter every slot, regardless of the channel status. When one node's backoff reaches 0, it transmits a busy tone to inform its neighbors that it has won the first stage. All nodes that hear this busy tone remain frozen in stage 1 until the next transmission commences. While the first stage occurs in parallel to the data transmission, the second stage begins at the end of the current transmission cycle. All nodes that enter the second stage wait for a second backoff interval before transmitting. If the channel remains idle until the end of the backoff duration, the node transmits its packet. If the channel is sensed busy before the end of the backoff interval, the node loses the contention, returns to the first stage, and recontends. If collisions occur in the second stage, the contention windows in both stages are exponentially increased. The end of the second stage is marked by a successful transmission. In this way, much of the protocol's idle overhead is effectively masked. The authors show that, at high loads, unlike IEEE 802.11, the saturation throughput remains close to the peak value. PPS also maintains the same level of fairness IEEE 802.11 displays.

Further analysis in [15] shows that the performance achieved is not sensitive to the probability of busy tone detection. In fact, even when a busy tone is not transmitted, the performance of PPS does not degrade significantly. Using this fact, the authors extend the PPS algorithm in [16] to propose the Dual Stage Contention Resolution (DSCR) protocol. DSCR has the same protocol structure but does away with the need for the busy tone channel. In DSCR, all deferring nodes, excluding the node that just transmitted the successful packet, decrement their first stage backoff counter by a fixed value F at the end of a successful packet transmission. After this, nodes decrement their backoff counters for each idle slot sensed. Nodes enter stage 2 when their backoff counter in stage 1 is less than or equal to 0. If nodes enter stage 2 at the beginning of a transmission cycle, they wait a random backoff interval before attempting to transmit. If a node enters stage 2 during a transmission cycle, it immediately commences transmission. Once a transmission commences on the channel, the other contending nodes return to stage 1, double their CW sizes, and recontend. Simulation results in [16] show that DSCR

achieves a higher performance than IEEE 802.11 and MACAW, even in the presence of hidden terminals. DSCR also uses a smaller minimum value for the contention window in stage 2. This increases the difference in CW sizes between the winning node and the rest of the network, similar to the situation seen in [7]. As a result, DSCR performs worse than IEEE 802.11 with respect to fairness. [16] shows that if the value of F is adaptively modified, so that nodes that have been in stage 1 longer decrease their backoff counters at a faster rate, the fairness of DSCR is significantly improved, though the performance gain is reduced.

Yang and Vaidya [17] compare PPS, DSCR, and the explicit pipelining scheme, similar to *DCMAC*. As the authors noted, the explicit pipelining scheme is only beneficial if both stages in the pipeline are balanced. This balance may be too difficult to achieve when pipelining IEEE 802.11 as the performance is expected to be heavily dependent on the bandwidth division, the distribution of the packet size, and the number of flows. Here, we implement an explicit pipelining scheme and evaluate the exact dependence of the performance on these factors. This will enable us to conclusively state if pipelining IEEE 802.11 is as impractical as it intuitively seem.

Deng et al. [18] explore the viability of pipelining the pure ALOHA protocol. Using results derived in [19], the authors derive the *pdf* of the contention resolution cycle on the control channel and then use it to show that there is always a nonzero waiting time on the data channel between transmissions. The waiting time is difference in duration between the control resolution cycle and the data transmission cycle. Using that result, they go on to prove that for pure ALOHA, the optimum throughput achieved for the in the pipelined case is always less than the optimum throughput achieved in the single channel case. In this thesis, we apply the explicit pipelining structure to IEEE 802.11 DCF to investigate if pipelining would result in any performance gains when applied to CSMA-based protocols.

CHAPTER 4

DCMAC PROTOCOL DESCRIPTION

4.1 Overview

The Dual Channel MAC (*DCMAC*) protocol uses the concept of pipelining to mask most of the overhead generated by the contention resolution process, so as to enhance the overall channel utilization in a WLAN. The total bandwidth is divided into a control channel and a data channel. *DCMAC* is divided into a contention-resolution stage and a data transmission stage. Contention resolution, via the use of the RTS/CTS mechanism, is performed over the control channel. Upon winning the right to transmit, the sender and the receiver switch over to the data channel to do the actual Data/ACK transmission. *DCMAC* adapts the IEEE 802.11 MAC protocol to explicitly support pipelining. The protocol also assumes that each node has one transceiver and hence is only able to transmit or receive over one channel at any instant.

Figure 4.1 illustrates how the principle of pipelining is employed across the control and data channels. As seen from the figure, while the $Data_1$ is being transmitted on the data channel, the remaining nodes stay on the control channel and commence the contention (RTS_2/CTS_2) for the next data packet, $Data_2$, that will be transmitted once the $Data_1/ACK_1$ dialog has completed. Thus, while one data transmission is ongoing on the data channel, contention resolution for the next data packet is simultaneously occurring on the control channel. In this way, the protocol is able to effectively mask the overhead

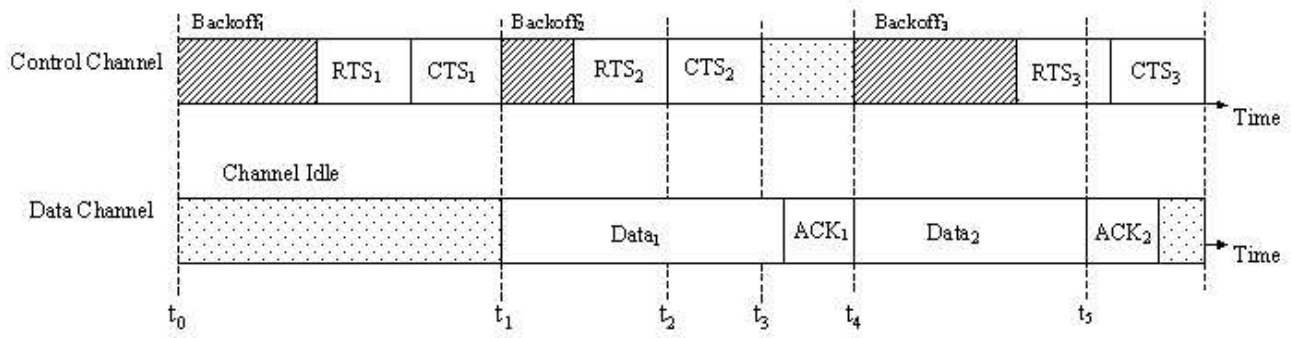


Figure 4.1 Overview of pipelining.

generated by the contention-resolution process. Ideally, each contention-resolution cycle should complete before the current Data/ACK transmission on the data channel completes so as to ensure maximum utilization of the data channel.

The *DCMAC* protocol is currently designed for WLANs. As such, the following descriptions are based on an isolated WLAN, where all nodes are within the transmission range of each other. The impact of relaxing the assumption of the isolated WLAN is discussed at the end of this chapter.

4.2 Definition of Terms

We define the following terms for ease of exposition; they are further illustrated in Figure 4.2.

Contention resolution cycle, W – This is the interval from the time the current contention began until the successful nodes switch to the data channel to commence data transmission.

Data transmission cycle, T – This is the duration of the current Data/ACK transmission on the data channel.

Start_time – This state variable denotes the time the current data transmission cycle began.

End_Time – This state variable either contains the time that the current data transmis-

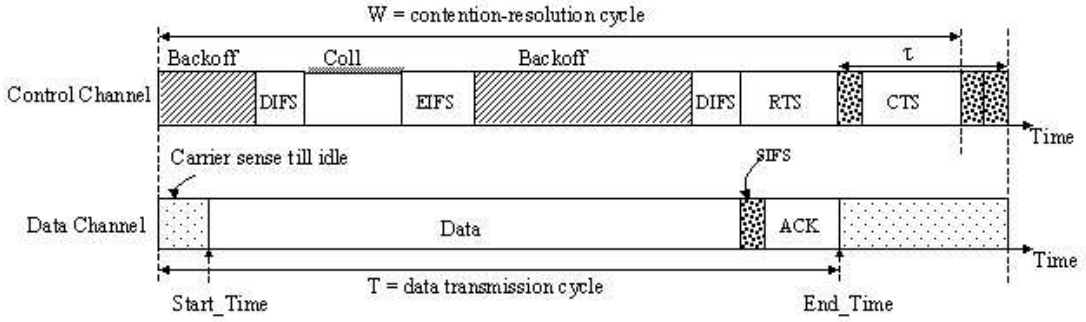


Figure 4.2 Illustration of terms.

sion cycle will end or denotes the time the most recent data transmission cycle ended.

Current_sender, Current_receiver – These state variables maintain the addresses of the sending and receiving nodes, respectively, that are currently on the data channel.

All nodes within the network maintain these four state variables so as to enable the correct execution of the MAC protocol.

4.3 RTS/CTS Packet Modification

Given that contention resolution and the data transmission occur on different channels, the duration for each cycle has to be advertised. Thus, both the RTS and the CTS control packets have an extra duration field added to them. The *requestTime* duration field advertises the interval that this sender and receiver pair expect to wait before switching to the data channel to begin transmission. The second duration field, *dataTime*, holds the duration of the corresponding Data/ACK exchange on the data channel. For example, from Figure 4.1, the packet RTS_2 will have its *requestTime* field set to $t_4 - t_2$ while CTS_2 will have its requestTime field set to $t_4 - t_3$. Both RTS_2 and CTS_2 will have their *dataTime* fields set to $t_5 - t_4$.

4.4 *DCMAC* Details

All nodes start on the control channel. Any node that wants to transmit must first carrier-sense the control channel. If the channel is sensed to be busy, the node defers its transmission until the channel is idle for at least a DIFS duration. Once the channel is sensed to be idle for at least DIFS, the node randomly selects a backoff counter value, b, from the uniform distribution over the interval [0,CW-1]. Similar to IEEE 802.11 DCF, a node must wait for b *idle* slots before it is allowed to transmit.

Once the backoff counter reaches 0, the node, say S, transmits an RTS to its destination node, say R, over the control channel. The *requestTime* field in the RTS contains the duration $t1$ that S expects to wait before it is able to switch to the data channel and begin transmission. The minimum value of $t1$ is $\tau = SIFS + CTS\_Time + 2 * SIFS$, which is the smallest duration that is required for the contention-resolution to complete and transmission to start (Figure 4.2). CTS_Time is the time required to transmit the CTS over the control channel. When R receives the RTS, it replies with a CTS, after an SIFS interval. R could have had a different duration $t2$ than it had expected to wait before switching to the data channel. So when replying, the *requestTime* field in R′s CTS is composed as follows:

$$CTS \rightarrow requestTime = max(t1 - CTS\_Time - SIFS,\ t2)$$

When S receives the CTS, it will wait the duration that R had stated in the CTS. This will ensure that both nodes S and R switch over to the data channel simultaneously.

Nodes other than S and R that overhear the RTS or the CTS use the duration fields and the address fields to update their own state. The address fields are used to update the *Current_sender* and *Current_receiver* fields while the nodes set their NAV on the control channel for the duration indicated in the *requestTime* field. This is to prevent collisions on the control channel, as nodes defer their transmissions when the NAV is set. The NAV is also used to prevent more than one flow within reception range of this RTS or CTS

from attempting to reserve the data channel for the next data transmission cycle. Once an RTS and a CTS have been exchanged on the control channel, it implies that the data channel has been reserved for the next transmission. As such, none of the other nodes is allowed to resume contending on the control channel until the next data transmission cycle commences. Thus, by setting the NAV to expire only when S and R switch to the data channel, the occurrence of multiple reservations is also prevented. The nodes also update *Start_time* to reflect the end of their NAV and use the *dataTime* field in the RTS or CTS to compute *End_time*, both of which are used to track the transmission on the data channel. The updating of the *Start_Time* and *End_Time* is done as follows:

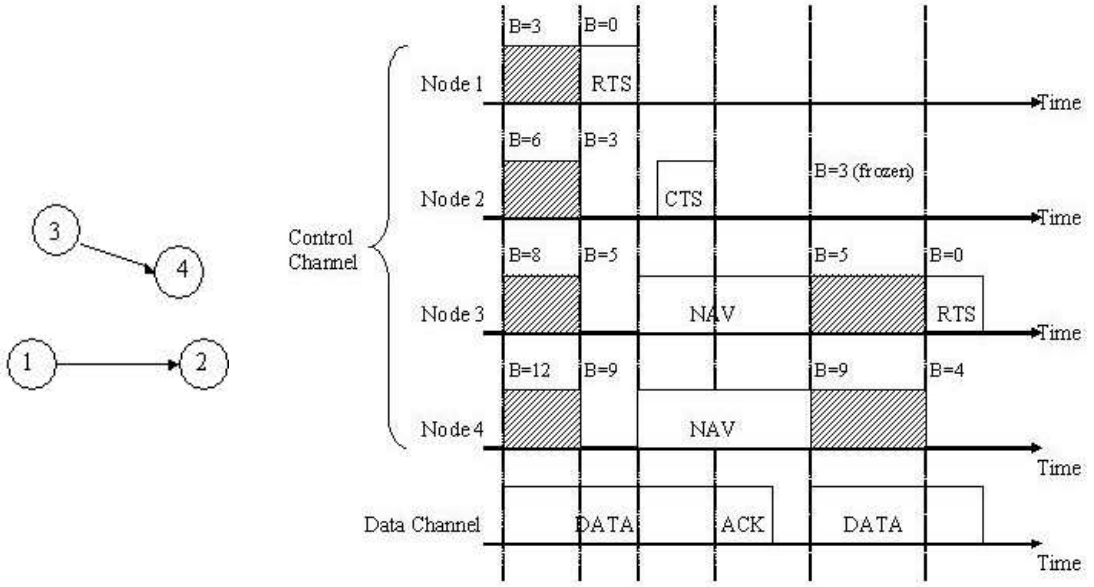
$$Start\_Time = \text{Time at which the NAV expires}$$

$$End\_Time = Start\_Time + dataTime$$

If there already is an ongoing transmission on the data channel, these state variables are assigned their new values at the end of the current data transmission cycle. This ensures that the state variables $Start\_time, End\_time, Current\_receiver$, and $Current\_sender$ are always consistent with the current data transmission.

Assuming that nodes S and R win the contention on the control channel, they will switch to the data channel at the end of the current data transmission cycle. Instead of transmitting immediately, both nodes first carrier-sense the data channel. If the channel is sensed to be idle, the data transmission commences. Otherwise the nodes enter a busy-wait loop, where they continually sense the channel and begin transmitting only when the channel is sensed idle. Since we are currently working within an isolated WLAN, the assumption is that both the sender and the receiver should see the same channel conditions. This assumption does not always hold, and the impact of differing channel conditions is discussed in a later section.

Given that both S and R had maintained a state that was consistent with the system, the NAV of the other nodes would expire at the same time S begins transmission on



(a) Two flows in a simple network

(b) Packet transmission with *DCMAC*

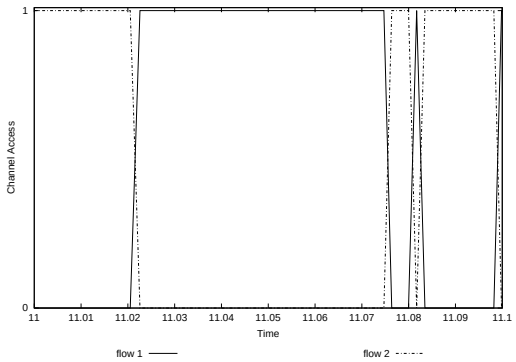
Figure 4.3 An example with *DCMAC*.

the data channel. At this point, these nodes can resume decrementing their backoff and contend for access rights for the next data transmission cycle.

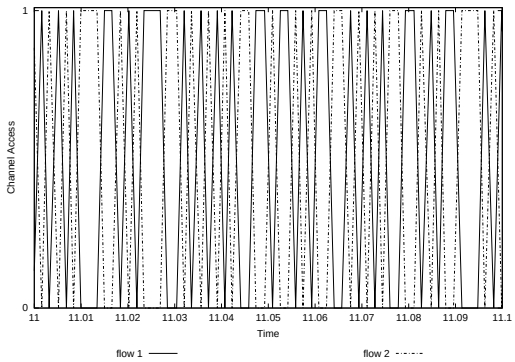
We illustrate *DCMAC* with an example. As shown in Figure 4.3(a), there is a flow from node 1 to node 2 and from node 3 to node 4. Also assume that both nodes 2 and 4 have backlogged packets to other nodes not shown in the Figure. As seen from Figure 4.3(b), node 1 is the first to decrement its backoff counter to 0 and hence sends an RTS to node 2. Upon sensing this transmission, all the other nodes freeze their backoff counters. When node 2 receives the RTS, it replies with a CTS. Since there is an ongoing data transmission, nodes 1 and 2 set their $requestTime$ fields to last until the end of the current data transmission, plus the extra duration needed to sense the data channel. Nodes 3 and 4 overhear both control packets and update their NAVs accordingly. At the end of the current data transmission cycle, nodes 1 and 2 switch to the data channel

and sense the medium. Since it is sensed as idle, the data transmission commences. At the same time, the NAVs on nodes 3 and 4 expire and they resume decrementing their backoffs. The backoff counter at node 2 remains frozen at its current value throughout the entire data transmission cycle as it is currently the receiver on the data channel.

The common receiver case requires special attention. The common receiver scenario occurs when at least two sources are sending packets to the same destination node. For example, both nodes 2 and 4 in Figure 4.3(a) are sending packets to a node 5 (not shown). In this case, if the flow from node 4 to node 5 won the channel access, both nodes will move to the data channel to commence transmission. Once node 4 starts transmitting, contention-resolution also resumes on the control channel. In the case where node 2 does not remember that node 5 is currently on the data channel, if it is the first to decrement its backoff value to 0, it will send out an RTS for node 5. Since node 5 is on the data channel, it will neither receive nor be able to respond to node 2's CTS on the control channel. As such, node 2 will not receive a reply to its RTS. In accordance with the protocol, the absence of a reply is taken to indicate that a collision occurred. This will cause node 2 to unnecessarily double its contention window and retry after a new random backoff delay. Consequently, the flow from node 4 to 5 will be able to capture the channel for an interval before node 2 is able to win the channel access. In the worst case, this could lead to an exponential increase the size of the node's CW and maybe even starve the flow from 2 to 5. Once node 2 wins the channel, it captures the channel until node 4 is able to steal it back. This behavior is illustrated in Figure 4.4(a). Fortunately, this phenomenon is easily prevented by using available information. Since all nodes maintain the state variables *Current_sender* and *Current_receiver*, if the address of a node's destination is equal to the address in either of these variables, the node freezes its backoff counter until this data transmission has finished before resuming its backoff. This not only prevents the exponential increase in the size of a node's CW, it also increases fairness between flows as channel capture by any individual flow is reduced, as shown in Figure 4.4(b).



(a) Short term channel capture.



(b) Channel capture circumvented via use of available information.

Figure 4.4 An example of how using available information regarding the current sender and receiver reduces channel capture. A value of 1 on the *Channel Access* axis is an indicator that the flow currently has access rights to the data channel.

As with any MAC protocol, collisions have to be accounted for. On the control channel, collisions occur when two nodes transmit an RTS at the same time. As in IEEE 802.11, collisions are detected by the absence of a CTS. When a collision is detected on the control channel, the colliding nodes double their contention windows and pick a new random backoff value. The other nodes that detect the collision set their NAV to EIFS. Since the EIFS duration is used to protect an impending CTS, the value of EIFS has been modified to reflect the time needed for a CTS to be successfully transmitted. If a collision occurs on the data channel, as a result of differing channel conditions, the colliding nodes have to recontend for the access rights and hence return to the control channel, double their CW, and pick a new backoff value.

Unlike the IEEE 802.11, *DCMAC* only works with the RTS/CTS access method. All data transmissions must be precluded by an RTS/CTS exchange on the control channel so as to reserve the access rights to transmit over the data channel. For broadcasts, the node wishing to broadcast a packet has to send an RTS on the control channel before it can broadcast the packet on the data channel. The RTS is to inform all nodes in the network about the pending broadcast. When a node receives an RTS for a broadcast, it

will switch over to the data channel at the end of the current transmission to receive the broadcast packet.

4.5 Discussion of Issues Arising from *DCMAC*

In this section, we identify and discuss the various problems that arise from performing the medium access resolution over two separate channels, instead of the single channel implementation adopted in the IEEE 802.11 standard.

4.5.1 Deafness

One of the most serious issues concerning *DCMAC* is deafness. Deafness, as identified in [20], is the phenomenon where some nodes are unaware of the network state because they are unable to receive the control packets that advertise the updates, i.e., the RTS and CTS control packets. *DCMAC* introduces deafness into even WLANs, where all nodes are within the transmission range of each other. In *DCMAC*, deafness arises as a result of a node's inability to listen on both channels simultaneously. For example, in Figure 4.3(a), when nodes 1 and 2 switch to the data channel to commence the data transmission, both the nodes will be deaf to any transmissions that are simultaneously occurring on the control channel. So, if the RTS node 3 sends to node 4 (Figure 4.3(b)) is successful, node 3 would have effectively reserved the channel after the current transmission. But both nodes 1 and 2 will not be aware of this. If, upon returning to the control channel, node 1 successfully wins the access rights to the data channel again, it will assume that the data channel should be free and immediately switch channels. In this case, the requirement that the channel has to be sensed as idle will prevent a collision on the data channel. So nodes 1 and 2 enter the busy-wait loop. Since the nodes remain on the data channel while in the busy-wait loop, they are again deaf to the state updates that are transmitted over the control channel. On the other hand, if another flow wins the channel contention, the 'deaf' nodes can use the information in the RTS and CTS packets to resynchronize their state variables to the actual state of the

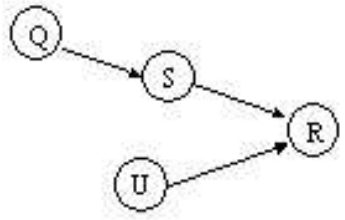


Figure 4.5 Deafness in the common receiver case.

network and thus avoid the above scenario. The probability that a flow consecutively captures the channel will decrease as the number of flows increases. As a result, the impact of deafness also becomes less pronounced as the number of flows in the network increases.

Deafness also affects the common receiver case. As discussed in Section 4.4, maintaining state variables reduces the effects of deafness on such cases. But in the case that a node is both a source and a destination, the impact of deafness is not mitigated. Using the network in Figure 4.5, assume that nodes Q and S are currently on the data channel. While that transmission is ongoing, node U wins the contention for the next data transmission. Once nodes Q and S are done, nodes U and R switch to the data channel. Now, since S was deaf to the state updates during the period it was receiving data from Q, it does not know that node R is on the data channel and hence will transmit an RTS. Since no reply can be forthcoming, S assumes a collision has occurred and doubles its CW before retrying. In the limit, this could result in the exponential increase of node S's contention window and could potentially starve the flow from S to R.

One possible solution to overcome all effects of deafness would be to outfit each node with an extra wireless transceiver. This will allow nodes to be able to listen to the transmissions on both channels, regardless of which channel they are on. In this case, the above scenario will not occur as S will be aware that R will be involved in the next scheduled data transmission and hence S can freeze its backoff counter until R returns

to the control channel.

The performance impact of the second transceiver will probably be most significant in networks with an access point present. These topologies are such that all the nodes in the network communicate with a single receiver i.e., an access-point. Whenever a data transmission is ongoing, all other nodes will freeze their backoffs since the common receiver is known to be on the data channel. In such scenarios, the benefits of pipelining are totally lost since the contention resolutions and the data transmissions become sequentialized. In fact, there is a significant performance degradation since the bandwidth allocated to the data channel is less than the total system bandwidth. With a second transceiver at the access point, the nodes will not have to freeze the contention-resolution process on the control channel since the receiver is able to participate in the scheduling process while receiving data on the control channel. Adding the extra transceiver enables such a topology to reap the benefits of the pipelined scheme, allowing a significant performance increase. The impact of an additional transceiver is not evaluated in this thesis, and is reserved for future work.

4.5.2 Differing channel conditions

DCMAC currently assumes that all nodes on a single channel sense the same channel conditions, especially since the network is operating as a WLAN. While this assumption may hold for an isolated WLAN, the presence of other WLANs or interference sources could result in different nodes sensing different channel conditions within the same wireless LAN. This impacts the transmissions on both the control and data channels.

On the control channel, a node may not reply to an RTS if it senses the channel to be busy, regardless of whether the packet was correctly received. This could again result in the sender's CW exponentially growing unnecessarily. Nodes would also freeze their backoff counters when they sense the channel to be busy. So if the interference source continually affects only a fixed set of nodes, then the rest of the nodes have a higher

probability of winning the contention. This is true assuming that the destination of the unaffected node also does not sense the interference.

On the data channel, differing channel conditions could cause either the sender or receiver to enter the busy-wait loop while the other node either commences transmission or expects an incoming transmission. If the sender enters the busy-wait loop, the receiver could time out and return to the control channel before the sender is able to commence transmission. As such, the data transmission will be unsuccessful and the sender will have to return to the control channel and initiate the retransmission process. If the receiver enters the busy-wait loop, it will still be able to receive the sender's transmission. Depending on how strong the interference is, the packet may or may not have been corrupted. If the packet is not corrupted, then the transmission will be successful as no carrier sensing is done before the transmission of an ACK. If the packet was corrupted, the receiver does not reply with an ACK and returns to the control channel. Consequently, it is still possible for the transmission to be unsuccessful and result in retransmission.

This can also be extended to multihop networks. The spatial distribution of the various flows generally results in varying channel conditions. For the case of the WLAN, the interference was a result of an external source situated at close proximity and affected a certain subset of the nodes. In a multihop network, the flows themselves act as the sources of interference for other flows. As such, the channel conditions across the network are constantly varying and dependent on the state of each flow. This also gives rise to the hidden-terminal problem, which results from the sender's inability to detect the channel conditions at the receiver. Thus, the severity of the performance loss would be worse in multihop networks.

Thus, differing channel conditions could seriously degrade the throughput achieved by increasing the duration of the average contention-resolution cycle and by increasing the

number of unsuccessful transmissions on the data channel.

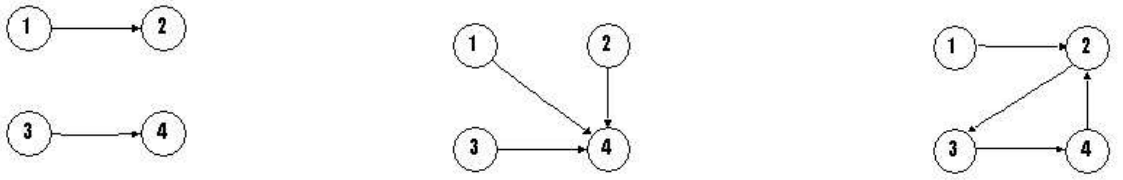
CHAPTER 5

PERFORMANCE EVALUATION

In this chapter, we study the performance characteristics of *DCMAC* in various network topologies via simulations. The simulations were done on a modified version of the *ns2* network simulator from USC/ISI/LBNL [21] and each data point is an average of 10 runs. The physical layer used is the direct-sequence spread spectrum (DSSS) and the parameters for DSSS, as defined in [2], are given in Table 5.1. The total bit rate is set to 11 Mbps, which is then divided between the two channels *DCMAC* requires. The physical header sizes were scaled appropriately so as to account for the 192-μs overhead incurred by transmitting the physical headers at the base rate of 1 Mbps. The carrier-sense mechanism was modified to include a one slot duration as the interval needed to determine whether or not the channel is idle. To maximize the channel utilization, all flows adopt an aggressive sending rate such that each always has backlogged packets to be sent. All the networks simulated are WLANs, i.e., every node is within transmission range of every other node in the network. The traffic sources are chosen to be constant

Table 5.1 DSSS specifications

SlotTime	20 μs
SIFS	10 μs
DIFS	50 μs
CW_{min}	31
CW_{max}	1023



(a) Disjoint flows (b) *AP* flows (c) Random flows

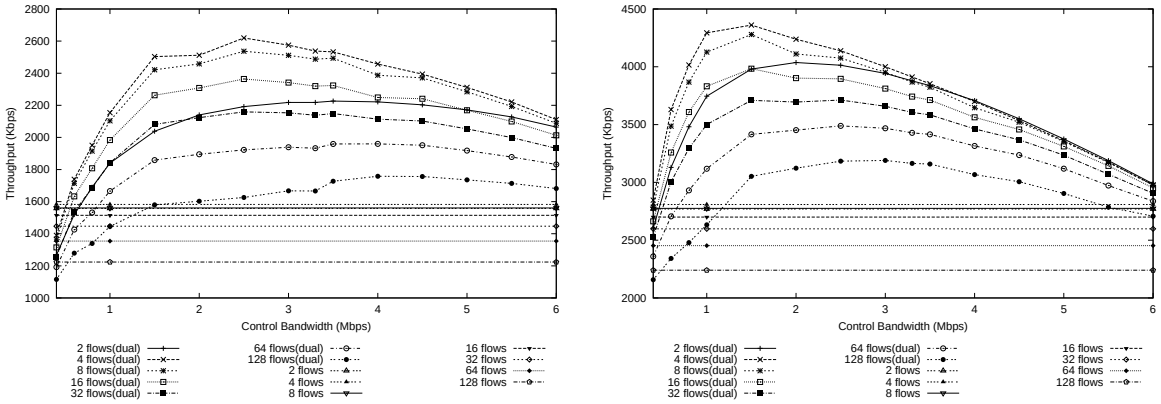
Figure 5.1 Example of the three different scenarios simulated.

bit rate (CBR) sources.

Three different types of network topologies were simulated. The first type is the disjoint flow scenario, where each node is either the sender or the receiver in exactly one flow. The second type is the access point (AP) flow scenario, where one node is the receiver for all the flows within the network. Finally, the random flow scenario is also simulated. Here, each node is the source for exactly one flow and its destination node is randomly selected from the rest of the nodes within the network. The three different scenarios are illustrated in Figure 5.1 for a four-node network.

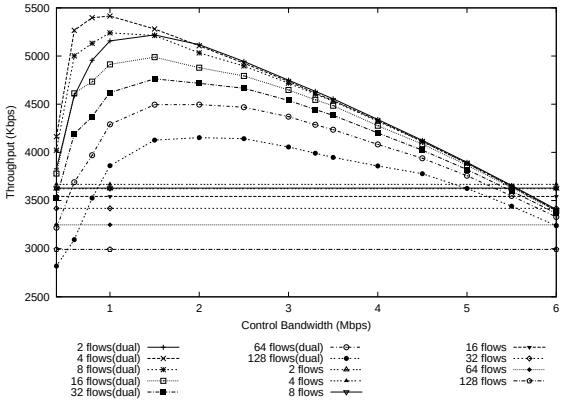
The simulation results for the disjoint flows are given in Figure 5.2. For all the graphs, the x-axis represents the bandwidth[1] allocated to the control channel, in Mbps, while the y-axis represents the aggregate throughput achieved, in Kbps. The key is given below each graph, where "2 flows(dual)" refers to the simulaton of *DCMAC* for a two-flow network and "2 flows" refers to the simulation of IEEE 802.11 for a two-flow network. For a given packet size, each figure shows the performance of both *DCMAC* and IEEE 802.11 DCF for varying number of flows. It can be seen that, across all packet sizes and number of flows, the throughput for *DCMAC* increases until a maximum and then decreases when the bandwidth assigned to the control channel is increased. Intuitively, the data channel

[1]The term bandwidth is used to loosely refer to the the bit rate.

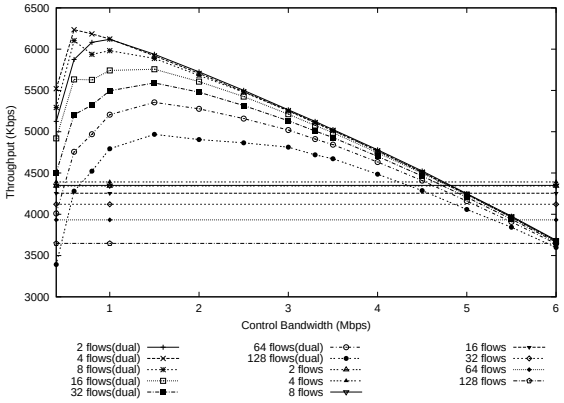


(a) 250 bytes

(b) 512 bytes



(c) 750 bytes



(d) 1000 bytes

Figure 5.2 Aggregate throughput versus control bandwidth for disjoint flows in a WLAN.

utilization, and hence throughput, is maximized when the contention-resolution for the next packet completes exactly at the end of the current data transmission cycle. This ensures that the idle time between data transmissions is minimized. This principle of balancing the stages is widely recognized as the basis of obtaining a performance gain in pipelined systems [22]. If the contention-resolution cycle is significantly longer than the data transmission cycle, it signifies that the bandwidth allocated to the control channel is insufficient and, as a result, the contention-resolution cycle is significantly longer than the data transmission cycle. The data channel utilization is affected by the longer idle

periods between data transmissions and the throughput also suffers as a result of the bottleneck on the control channel. In this case, increasing the control channel bandwidth eases the bottleneck on the control channel and increases the system throughput. On the other hand, if the data transmission cycle is significantly longer than the contention resolution cycle, it implies that the bandwidth division is again suboptimal, causing the data to take a very long time to transmit. In such cases, while the utilization of the data channel is maximized, the achievable throughput is limited by the small bandwidth assigned to the data channel. Decreasing the control channel bandwidth increases the throughput since the bottleneck on the data channel is reduced. Thus, the peak throughput occurs at the bandwidth division that approximately balances the contention resolution cycle with the data transmission cycle.

As the packet size increases, the optimum bandwidth division occurs at smaller control channel bandwidths, regardless of the number of flows present in the network. For data packets of 250 bytes, the optimum control channel bit rate occurs in the range of 2.5 to 3.5 Mbps (i.e., approximately 23% to 32% of the total bit rate). For packet sizes of 1000 bytes, this range shifts left to 0.5 to 1.5 Mbps (i.e., 5% to 14% of the total bit rate). This shift is due to the fact that as the packet size increases, the packet transmission time also increases. As such, more bandwidth can be allocated to the data channel to reduce the absolute value of the data transmission cycle. With longer data transmission cycles, the duration of the contention-resolution cycles can be allowed to increase proportionately by decreasing the bandwidth allocated to the control channel. Thus, the equilibrium point occurs at a smaller control channel bandwidths.

The aggregate throughput curves tend to converge as the control bandwidth increases. This is expected because, as a larger fraction of the bandwidth is allocated to the control channel, the data transmission cycle becomes the bottleneck in the system. Furthermore, as the packet size increases, the duration of the data transmission cycle also increases, which causes the bottleneck on the data channel to become more pronounced. This is

evident from the way the graphs in Figures 5.2(a) - 5.2(d) begin to converge at smaller control channel bandwidth values for larger packet sizes.

For a fixed packet size, as the number of flows in the network increases, the optimal bandwidth division occurs at larger control bandwidth values. For example, given a packet size of 512 bytes, the peak throughput for 4 to 32 flows occurs around a control channel bandwidth of 1.5 Mbps, while the peak throughput for 64 and 128 flows occurs at a control bandwidth of 2.5 Mbps. This is because the number of collisions on the control channel increases with the number of flows and, as a result, the average contention-resolution duration increases. Consequently, the control bandwidth has to be increased in order to maintain the balance between the contention-resolution cycle and the data transmission cycle.

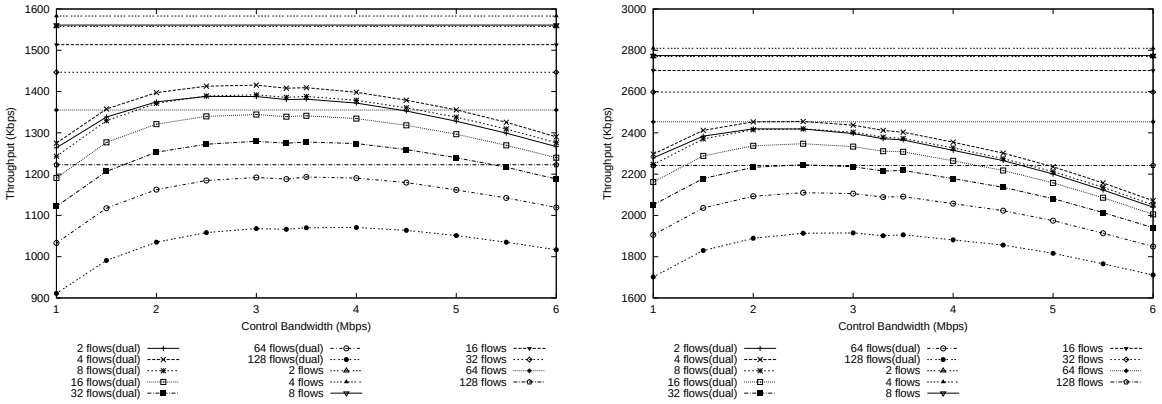
The two-flow scenario is an exception to the above trend as its peak throughput occurs at a larger bandwidth division than for the four-flow scenario. For two flows, there is usually only one flow on the control channel. As a result, the expected idle period is 16 slots.[2] As the number of flows increases, the probability that at least one flow has a smaller backoff value increases and thus the expected idle period is also smaller. The reduction in the idle overhead reduces the contention-resolution cycle, and consequently the optimal bandwidth division occurs at a smaller control channel bandwidth. But as the number of flows increases, the larger number of collisions results in longer contention-resolution cycles and thus causes the optimal bandwidth division to shift right.

Figure 5.2 also compares the performance of *DCMAC* with that of IEEE 802.11 DCF. In general, *DCMAC* performs better than IEEE 802.11 DCF since *DCMAC* masks the overhead introduced by the contention resolution, particularly the idle duration that is a result of the random backoff procedure, with the ongoing data transmission. But, as the control bandwidth increases, the performance of DCMAC eventually becomes worse than

[2]This is the expected backoff value with the minimum contention window of 31 slots.

IEEE 802.11 because the system throughput is restricted by the small bandwidth allocated to the data channel. This is seen from the graphs where the *DCMAC* performance curves drop below those of IEEE 802.11. This performance degradation becomes more pronounced as the packet size increases. From Figures 5.2(a) and 5.2(b), it is evident that the performance of *DCMAC* is always better than IEEE 802.11, regardless of the number of flows. As the packet size increases, the performance of *DCMAC* becomes worse than IEEE 802.11 at progressively lower control bandwidths (Figures 5.2(c)-5.2(d)). This effect is more obvious for a smaller number of flows. For example, the performance of *DCMAC* in a network with four flows degrades below that of IEEE 802.11 around a control bandwidth of 5.5 Mbps, given a packet size of 512 bytes. When the packet size increases to 1000 bytes, beyond a control bandwidth of about 4.7 Mbps, IEEE 802.11 performs better than *DCMAC*. But when there are 128 flows in the network, the performance of *DCMAC* only degrades below that of IEEE 802.11 at around 5.9 Mbps and only when the packet size is 1000 bytes. But this degradation occurs very far from the optimal bandwidth division. Thus, overall, *DCMAC* performs significantly better than IEEE 802.11 for disjoint flows.

Figure 5.3 shows the results of simulating the AP flow scenario in a WLAN for a range of flows and packet sizes. The general performance characteristics for *DCMAC* are similar to that of the disjoint flows. The throughput peaks at a certain bandwidth division and, as the packet size increases, this optimum bandwidth division occurs at smaller control channel bandwidths. Regardless of the number of flows present in the system, the aggregate throughput the system achieves also tends to converge as the control bandwidth increases. While the general performance characteristics of both scenarios are the same, the absolute throughput achieved by *DCMAC* in the AP flow scenario is significantly smaller than that of the disjoint flows and is always worse than IEEE 802.11 DCF. This is due to the presence of the access point, which is the common receiver across all flows in the network. Given that *DCMAC* freezes a node's backoff counter if its destination is known to be on the data channel, the contention-resolution stage is frozen during



(a) 250 bytes

(b) 512 bytes

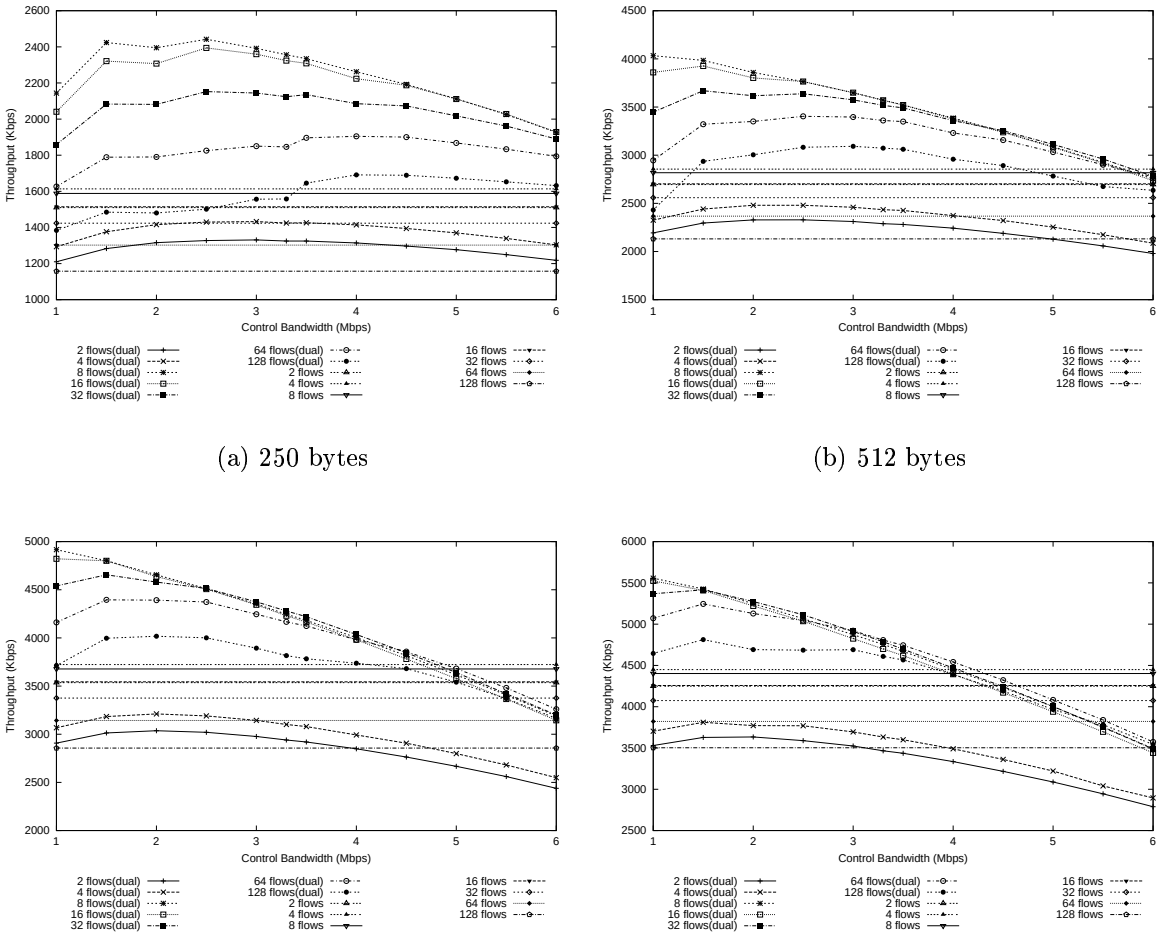
(c) 750 bytes

(d) 1000 bytes

Figure 5.3 Aggregate throughput versus control bandwidth for the access-point topology in a WLAN.

every data transmission cycle. This effectively sequentializes the contention-resolution stage and the data transmission stage. Consequently, only one channel is in use at any instant. This bandwidth wastage reduces the throughput that the system is able to achieve. Furthermore, since the data channel bandwidth is always less than the total available bandwidth, *DCMAC* will not be able to match the performance of IEEE 802.11 DCF in the AP scenario. The *DCMAC* throughput in this scenario can be improved by using two transceivers at the access point, which will allow pipelining to proceed while

it is receiving on the data channel.



(a) 250 bytes

(b) 512 bytes

(c) 750 bytes

(d) 1000 bytes

Figure 5.4 Aggregate throughput versus control bandwidth for the random topology in a WLAN.

Unlike the disjoint flows and the AP flows, random flows are more realistic as they account for the possibility that a node could be both a sender as well as a receiver. This interdependence will affect the aggregate throughput as deafness becomes an important factor. The issue of deafness was discussed in Section 4.5.1. The results for the random flow simulations are given in Figure 5.4. While the performance curves for the random flows follow the same trends as those of the disjoint and AP flows, the throughput

achieved by *DCMAC* for random flows lies within the range bounded by the throughput of the disjoint flows and that of the AP flows. As described earlier, the random flows were picked such that the each node in the network is a source of exactly one flow and its destination is randomly selected from the remaining nodes in the network. For a small number of flows, most of the flows become interdependent. Two flows in a network are independent if there are no common nodes between them; otherwise they are interdependent. Consequently, the contention resolution and data transmission stages become heavily sequentialized. The network becomes very similar to that of the AP flows and hence the performance of *DCMAC* is worse than IEEE 802.11 DCF for a small number of flows.

As the number of flows increases,[3] there are a larger number of flows that are independent of each other. With a greater number of independent flows, the impact of pipelining becomes more obvious. When one flow is on the data channel, all the flows that are dependent on this flow will freeze their backoff counters and not participate in the current round of contention. Flows that are independent of the nodes currently on the data channel proceed with the contention-resolution on the control channel. There is a high likelihood that, by the end of the current data transmission cycle, one of the independent flows would have won the contention-resolution on the control channel and would be ready to switch to the data channel to start transmitting. Thus, the presence of independent flows within a network causes the random flow scenario to behave like the disjoint flow scenario, thus resulting in an increase in the system throughput. This is evident from the sharp increase in system throughput from a four-flow scenario to an eight-flow scenario. As the number of flows continues to increase, the throughput gain that results from the presence of independent flows within the network is slightly offset by an increase in the number of collisions. Despite this, the presence of independent flows within a network still contributes to a significant performance gain. As seen from

[3] As mentioned earlier, each node is the source of only one flow. Hence, the number of nodes in the random flow network increases with the number of flows.

Figure 5.4, the performance of a network with 128 flows is always better than that with two or four flows, regardless of the data size.

The performance of *DCMAC* degrades below that of IEEE 802.11 faster for random flows than it does for disjoint flows. This is indicative of the impact the interdependence and deafness have on the overall system throughput. For a packet size of 750 bytes, the performance of *DMCAC* for an eight-flow network becomes worse than that of IEEE 802.11 as the control bandwidth becomes larger than 5 Mbps. In contrast, *DCMAC* performed better than IEEE 802.11 up to a control bandwidth of 5.5 Mbps for an eight-flow network in the disjoint flow scenario (Figure 5.2(c)). This implies that the performance gain is more sensitive to the bandwidth division in the random flow scenario than in the disjoint flow scenario.

From the above discussion, it can be inferred that, for a random network topology, the benefit of *DCMAC* is dependent on the number of flows and the packet size distribution as well as the bandwidth allocation between the two channels. Despite the fact that the maximum throughput occurs at a certain bandwidth division, the decrease in the aggregate throughput for a bandwidth division around the optimum is not very significant. This gradual degradation implies that, given the packet and flow distributions, a certain bandwidth division can be chosen such that the system throughput across all number of flows is close to the peak throughput. Hence, the improvement of *DCMAC* over IEEE 802.11 is more sensitive to the number of flows in the network and to the packet size distribution than it is to the bandwidth division.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we describe and evaluate the Dual Channel MAC (*DCMAC*) protocol that incorporates an explicit pipelining structure. *DCMAC* adapts the IEEE 802.11 DCF protocol by splitting the transmission dialog into a contention-resolution stage and a packet transmission stage. The protocol divides the available bandwidth between two channels, namely a control channel and a data channel. The contention-resolution stage occurs over the control channel and involves the RTS/CTS exchange to win the access rights to transmit the data packet. Upon winning the contention-resolution, the DATA/ACK exchange is done on the data channel. By allowing the contention-resolution for the next data packet to occur in parallel with the current data transmission, a significant portion of the contention overhead is masked and thus results in a throughput gain.

We simulated the *DCMAC* protocol within an isolated WLAN while varying the packet size, bandwidth division, and number of flows. As expected, there is an optimum bandwidth division such that the duration of the contention-resolution stage balances the duration of the data transmission stage, and the aggregate throughput is maximum at this bandwidth division. The control bandwidth at which this optimum bandwidth division occurs increases with the number of flows. Furthermore, as the bandwidth on the data channel decreases, the aggregate throughput that can be achieved tends to converge to a single value, regardless of the number of flows and the packet size. These trends hold across all types of flow topologies. *DCMAC* results in a substantial performance

improvement over IEEE 802.11 DCF for disjoint flows, but it performs significantly worse for the AP flow scenario. The random flow scenario is a middle point between these two scenarios and accounts for the common receiver problem as well as the impact of deafness on the overall performance. From the results, it is evident that the performance is more sensitive to the distribution of packet size and to the number of flows than it is to the bandwidth division. The optimum bandwidth division for all number of flows occurs within a range of 1 Mbps of each other. Also, a slight deviation from the optimum bandwidth allocation does not significantly degrade the throughput of the system. Hence, if the packet size distribution is known, a fixed bandwidth division can be selected such that the system throughput across all number of flows is relatively close to the peak value.

The results presented in Chapter 5 show that *DCMAC* does improve on the performance of IEEE 802.11 in most cases. The next step would be to attempt to extend the pipelining protocol to the multihop scenario and evaluate the actual impact differing channel conditions have on the aggregate throughput. Another possible direction for future work would be to outfit each node with two transceivers. The evaluation in the thesis was done under the assumption that each node only has one transceiver and is only able to send or receive on one channel at any point in time. As such, dependent flows become sequentialized since contention-resolution cannot proceed while the common node is on the data channel. With two transceivers, nodes will be able to send and receive on both channels simultaneously. Consequently, a node will be able to participate in the contention-resolution process on the control channel while it is involved in the current transmission on the data chanel. This will solve both the common receiver problem and the deafness issue.

DCMAC currently only supports the RTS/CTS access method. All transmissions have to be preceded by an RTS/CTS exchange while broadcasts have to be preceeded by an RTS on the control channel. As a result, for small packets, the overhead becomes a very significant portion of the transmission cycle. Recognizing this, IEEE 802.11 has a certain

predetermined threshold, and for packets smaller than this threshold, the basic access method is preferred over the RTS/CTS access method. *DCMAC* currently does not accommodate for this. One possible solution would be to allow nodes to transmit packets smaller than some threshold size directly on the control channel instead of having to be preceeded by the RTS/CTS exchange. The threshold size will represent the smallest packet size for which the overhead of the RTS/CTS exchange is still acceptable. Futhermore, this threshold value will vary with the bandwidth division since the overhead of the RTS/CTS exchange is depenent on the control channel bit rate. A similar scheme could also be adopted for broadcast packets. Modifying *DCMAC* to accomodate these nuances will be an important part of our future work.

REFERENCES

[1] R. Jain, G. Babic, B. Bagendra, and C. Lan, "Fairness, call establishment latency and other performance metrics," ATM Forum/96-1173, ATM Forum Document, Tech. Rep., August 1996.

[2] IEEE 802 LAN/MAN Standards Committee, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," IEEE Standard 802.11, 1999.

[3] A. S. Tanenbaum, *Computer Networks*, 4th ed., Redwood City, CA: Prentice-Hall, 2003.

[4] L. Kleinrock and F. A. Tobagi, "Packet switching in radio channels: Part I - Carrier sense multiple access modes and their throughput-delay characteristics," *IEEE Transactions on Communications*, vol. COM-23, no. 12, pp. 1400–1416, 1975.

[5] P. Karn, "MACA - A new channel access method for packet radio," in *Proceedings of 9th ARRL Computer Networking Conference*, 1990, pp. 134–140.

[6] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[7] Y. Kwon, Y. Fang, and H. Latchman, "A novel MAC protocol with fast collision resolution for wireless LANs," in *Proceedings of IEEE INFOCOMM*, vol. 2, April 2003, pp. 853–862.

[8] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless lans," in *Proceedings of ACM SIGCOMM*, vol. 24, October 1994, pp. 212–225.

[9] F. Cali, M. Conti, and E. Gregori, "Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit," *IEEE/ACMTransactions on Networking*, vol. 8, no. 6, pp. 785–799, 2000.

[10] F. Cali, M. Conti, and E. Gregori, "IEEE 802.11 protocol: Design and performance evaluation of an adaptive backoff mechanism," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 9, pp. 1774–1786, 2000.

[11] ETSI TC-RES. "Radio equipment and systems(RES); HIgh PErformance Radio Local Area Network(HIPERLAN) Type 1; Functional specification," European Telecommunication Standard ETS 300 652, 1996.

[12] G. Anastasi, L.Lenzini, and E. Mingozzi, "Stability and performance analysis of HIPERLAN," in *Proceedings of IEEE INFOCOMM*, vol. 1, April 1998, pp. 134–141.

[13] J. Weinmiller, M. Schlager, A. Festag, and A. Wolisz, "Performance study of access control in wireless LANs - IEEE 802.11 DFWMAC and ETSI RES 10 HIPERLAN," *Mobile Networks and Applications*, vol. 2, no. 1, pp. 55–67, 1997.

[14] T. D. Todd and J. W. Mark, "Capacity allocation in multiple access networks," *IEEE Transactions on Communications*, vol. COM-33, no. 11, pp. 1224–1226, 1985.

[15] X. Yang and N. H. Vaidya, "Pipelined packet scheduling in wireless LANs," Coordinated Science Laborotary, University of Illinois at Urbana-Champaign, `http://www.students.uiuc.edu/~xueyang/mypaper/pipeline.ps`, Tech. Rep., August 2002.

[16] X. Yang and N. H. Vaidya, "DSCR: A more stable MAC protocol for wireless networks," Coordinated Science Laborotary, University of Illinois at Urbana-Champaign, `http://www.students.uiuc.edu/~xueyang/mypaper/dscr-tech.ps`, Tech. Rep., August 2002.

[17] X. Yang and N. H. Vaidya. "Explicit and implicit pipelining for wireless medium access control," Invited Paper, Vehicular Technology Conference, 2003.

[18] J. Deng, Y. S. Han, and Z. J. Haas, "Analyzing split channel medium access control schemes with ALOHA reservation," in *Ad-Hoc, Mobile, and Wireless Networks - ADHOC-NOW '03*, vol. 2865 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag, 2003, pp. 128–139.

[19] H. Takagi and L. Kleinrock, "Output processes in contention packet broadcasting systems," *IEEE Transactions on Communications*, vol. COM-33, no. 11, pp. 1191–1199, 1985.

[20] R. Roy Choudhury and N. H. Vaidya, "Deafness: A MAC problem in ad hoc networks when using directional antennas," Coordinated Science Laborotary, University of Illinois at Urbana-Champaign. http://www.crhc.uiuc.edu/%7Ecroy/pubs/cameraready.ps, Tech. Rep., July 2003.

[21] K. Fall and K. Varadhan, "Ns notes and documentation," Tech. Rep., UC Berkley, LBL, USC/ISI, Xerox PARC, 2002.

[22] J. Hennesey and D. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed., San Francisco, CA: Morgan Kaufmann Publishers Inc., 2002.