University of Alberta

A MAC Protocol for Multihop RP-CDMA Ad Hoc Wireless Networks

by

Todd Mortimer

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2012
Edmonton, Alberta

Abstract

RP-CDMA is a wireless multiple access technique that utilizes multiple spreading codes and a multiuser detector to enhance link reliability and performance. We propose a simple MAC protocol on top of the RP-CDMA Phy and apply it to the multihop ad hoc network model. In addition to a MAC, we propose two significant extensions to RP-CDMA aimed at improving throughput and reliability. We test our network device using the ns-3 network simulator and compare its performance to that of the well known 802.11 CSMA model. Our simulations confirm that RP-CDMA can substantially improve link reliability and network performance, but that a link level acknowledgement mechanism is required to ensure packet delivery across the network. We investigate a simple acknowledgement policy and conclude that it enables simultaneous high throughput and packet delivery while maintaining low latency.

Acknowledgements

Foremost I would like to thank Dr. Janelle Harms for her guidance and patience, particularly when I was indulging my impulses. Thanks for letting me disappear into the MAC layer - I know we started out up higher. I would also like to thank Dr. Christian Schlegel for his feedback and suggestions as I recast RP-CDMA for my purposes. My conversations with Dr. Ivan Fair and Majid Ghanbarinejad were invaluable for understanding what's actually going on in a radio, and I am grateful for their time and assistance. Of course, I must also thank my wife, Erin, whose endless support in everything outside academics has made the academics themselves so much more manageable.

Finally, I would like to thank the Department of National Defence and the Canadian Forces, whose sponsorship has made all of this possible. I hope the investment in me proves fruitful when I return to service.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The proliferation of wireless data networks in the last decade has provided ever faster and more robust service. Since 1999, the popular 802.11 standard has progressed in data rate from an initial 1 Mbps to up to 600 Mbps in the current revision [49]. At the same time, fundamental limitations of the wireless channel have necessitated increasingly complex signalling protocols and increasing bandwidth allocations in order to support these ever faster services [39]. The primary limitation which all wireless systems must contend with is the shared nature of the wireless channel. Because any devices working in the same wireless channel must necessarily share it with other devices in the network, mechanisms and protocols for dividing up the available bandwidth among all of the network peers must be developed. The result is that each device necessarily has only partial access to the full channel capacity, and any errors in the coordination of channel sharing result in transmission collisions, lost data, and diminished link reliability.

This problem is particularly evident in ad hoc networks. An ad hoc network is one in which the network nodes join a common channel in the absence of any fixed infrastructure or network controlling authority [45]. These nodes build a logical network using the wireless channel and exchange routing information in order to cooperatively pass traffic across the network. This is in contrast to the more familiar base station model in which nodes join a network created and managed by a central network node. The central node is responsible for admitting each network peer and coordinating their transmissions so as to not collide.

We are specifically interested in the multihop ad hoc model here because of the

additional challenges it presents compared to the base station model. In the presence of a base station, network nodes need only coordinate their transmissions with the base station, which can strictly control the behaviour of all of the nodes and ensure that each node gets their share of the channel. Each node passes all of their traffic through the base station and the base station tells the nodes when to transmit and when to listen, so there is no need coordinate with any other node in the network. In an ad hoc network, individual network nodes must coordinate among themselves in order to pass messages between any of their peers. In the event that two nodes cannot communicate directly, a multihop ad hoc network will pass messages from node to node between the message originator and destination. Since the radio is a half duplex device, nodes must decide when to exclusively transmit and when to listen. Dividing up which nodes will transmit, to which other nodes, when they do so, and how their transmissions are sent turns out to be a complex operation in an environment which lacks any sort of central authoritative coordinating entity. The consequence of getting this coordination function wrong is data loss, which either results in delays as packets are retransmitted or outright loss as data simply disappears from the network. The ad hoc environment is therefore somewhat more challenging to work in, and presents unique problems that do not have obvious solutions.

There are many ways in which several peers might coordinate access to a common wireless channel. Perhaps the most obvious approach is to try to avoid transmitting at the same time as other nodes, which is the principle behind the popular 802.11 wireless standard [48]. Alternately, the available medium can be divided up into multiple subchannels, and nodes can coordinate access to these subchannels in a non-interfering way. Three of the most common ways to divide up the wireless medium are to set up a time schedule (TDMA), divide different peers into different non-interfering frequency groups (FDMA), or to use spread spectrum techniques such as code division to separate users into different code channels (CDMA) [23]. Each of these methods has their own drawbacks, which we will discuss in Chapter 2, but fundamentally they all suffer from the same limitation imposed by the shared nature of the wireless channel: Because nodes need to avoid collisions they

must organize multiple access to the channel, but this coordination must be done using the shared channel itself. The coordination traffic, therefore, is itself subject to collisions which causes the loss of both data traffic and coordination traffic. The loss of coordination traffic may cause further data loss as the access coordination function breaks down.

The root of this problem is the nature of the wireless channel itself and its unreliability due to collisions. Were the channel reliable such that multiple nodes could transmit simultaneously without having to coordinate channel access then we could dispense with the coordination function altogether. Without the need to coordinate channel access then the decision of whether an ad hoc network node transmits at any given time is simplified to deciding whether the intended receiver node is listening. Determining whether a neighbouring node is listening is the inverse of deciding whether they are transmitting, which can be simply estimated by listening to the channel itself.[1] With a reliable link it should therefore be possible for us to construct an ad hoc network in which nodes can decide whether or not to transmit without consulting any other nodes or external sources of information.

In this dissertation, we propose an ad hoc Medium Access Control (MAC) protocol built on top of such a reliable link, and evaluate its performance compared to traditional 802.11 Carrier Sense Multiple Access (CSMA). This reliable link was proposed by Schlegel *et al.*[42], and uses code division to separate individual packets into private channels, with the result that multiple packets can arrive simultaneously at a receiver with only a relatively small probability of colliding. We propose two extensions to the link itself, and on top of it we propose a simple MAC protocol which does not require any coordination between nodes. Our aim is to show that with a reliable link it is possible to build ad hoc networks which exhibit improved reliability, and hence performance, compared to networks built on top of traditional wireless links.

In the remainder of this chapter we motivate our work by first discussing the wireless link reliability problem in general and then identifying the characteristics

[1] We assume that nodes are continuously awake and do not have a sleep / wake cycle or some other mechanism which causes them to have periods in which nodes are neither transmitting nor listening.

of a theoretical wireless link that would address this problem. With the problem articulated, we identify a potential solution and then describe our approach to measuring and testing its effectiveness.

1.1 Wireless Network Reliability

We are interested here in a reliable wireless ad hoc network. Network reliability ultimately depends on link level reliability, with the successful routing of packets through the network being dependent on the successful passing of packets across each link along the way. As discussed above, the fundamental problem with wireless link reliability is the shared nature of the wireless channel.

If we take a moment to look back at wired Ethernet networks, we recall that these networks once also had shared channels with the use of hubs, and also experienced collisions and loss. In Ethernet, the collision problem was solved by switching from hubs to switches, which provided each node in the network with a private channel to the switch. With the widespread deployment of switched packet networks over interference-resistant cabling, reliable wired links have enabled ever faster wired networks and eroded the requirement for complex protocols at the MAC and link layers [50].

It is therefore natural to ask how we could achieve the same thing in a wireless network. It is possible to separate transmissions into separate channels using either time, frequency or code division, but the problem then becomes one of channel coordination. Which node gets what channel, when and for how long? In an ad hoc network a node may have a requirement to communicate with any or several of its neighbours, and must therefore coordinate with each of them in order to decide which time slot, frequency or spreading code to use for a given transmission. As more load is placed on the network and more nodes contend for the same channel resources, this distributed coordination of orderly access to the channel breaks down, resulting in declining link reliability and system performance.

Thus, in an ad hoc network, coordinating access to the shared wireless medium among disparate nodes poses a significant challenge, particularly as this coordina-

tion must be done using the wireless medium itself. This difficulty in coordinating the orderly access to the shared wireless medium has resulted in complex MAC and link layer standards for wireless networks [48], with limited success [56, 40]. Additionally, even if we could perform this coordination perfectly each node could still access only a fraction of the total channel capacity, since nodes must take turns transmitting and receiving.

Finally, the wireless channel must contend with external noise and interference. There may be many wireless devices in a given area, each of which contributes energy, and hence noise, to the wireless medium. Transmissions may reflect or scatter off of physical objects in the environment on their way to their destination, which can cause signal fading, multipath interference, or echoes, all of which must be suppressed or otherwise handled by the wireless receiver [32]. In contrast, wired networks are less susceptible to external interference, which can often be addressed simply with shielded cabling, and the properties of the wire can be controlled so as to mitigate against unrecoverable signal fading or distortion.

We see then that wireless network reliability is a different sort of problem from its wired counterpart. Nonetheless, we can look to the wired link to identify those properties which have enabled its rapid advancement in reliability and performance, and try to construct a wireless link that exhibits some or all of those properties.

1.2 Problem Description

Ideally, we would like wireless networks to be more like wired networks, with simple and reliable links and protocols. In a modern wired network, the wire provides each node with a private channel to their next hop, and is constructed so as to be resistant to external interference. Under these conditions, there is no requirement for nodes to coordinate access to the network, and so each node can transmit randomly and simultaneously up to the capacity of the link. Translating this to a wireless context, we desire a wireless link that holds the three qualities:

1. Each node has a private channel to each of their peers.

2. Signals are resistant to interference.

3. Each node can access the channel without coordinating with its peers.

We will see that each of these properties is already available in the ad hoc wireless link, but not always together with the others. For example, we can easily achieve private channels between peers by assigning each node a unique frequency for communicating with each of its neighbours, but in order to use those frequencies effectively the transmitting node must coordinate with an intended receiver about when to switch to the private channel [46]. Similarly, we can adopt spread spectrum techniques, such as code division, to overcome reasonable levels of external and internal interference, but must then coordinate between nodes for which spreading codes or subcarriers to use, lest the signals be missed and lost [22]. Finally, we can eschew coordination and have nodes simply broadcast when they have data traffic, as in ALOHA [3], but this means that all nodes must work in a single channel.

Our problem is therefore to identify a means of accessing the wireless channel that exhibits all three of these desirable properties simultaneously.

1.3 Aim

We shall see that a multiple access method, Random Packet Code Division Multiple Access (RP-CDMA) [42], has been proposed in the literature that comes very close to meeting all of our desirable properties. We propose to apply this method to the multihop ad hoc context and evaluate its reliability and performance, compared to the popular Carrier Sense Multiple Access (CSMA) 802.11 standard.

RP-CDMA is a type of CDMA protocol that uses multiple spreading codes to separate individual packets into private channels, where a channel is defined by its spreading code. It therefore provides private channels for each communication, and has the same interference-resistant properties of other CDMA protocols. By virtue of its packet format, RP-CDMA also offers completely uncoordinated and asynchronous channel access. It is these three qualities that make RP-CDMA a promising candidate for a reliable wireless link. Previous work has investigated the reliability and performance characteristics of RP-CDMA over single network hops [43, 16], and here we investigate its application in the general multihop ad hoc

context. Our aim is therefore to establish RP-CDMA as a suitable multiple access method for reliable, high performance, ad hoc networks.

1.4 Approach

In order to test our thesis that RP-CDMA can form the basis of a reliable wireless ad hoc link we undertook a performance study using simulation. To this end we implemented a simulated RP-CDMA transceiver, constructed a MAC protocol on top of it, and then tested the reliability and performance of multihop ad hoc networks using this network device. We evaluated our results both absolutely and in comparison to an 802.11 network device under the same conditions.

In the process of our performance study we developed algorithms which define the functioning of our proposed MAC protocol and identified parameters which may be adjusted to optimize system performance under particular conditions. By permuting each of these parameters throughout their range, we examined their effects on link reliability and system performance. In this dissertation we describe the algorithms which make up our proposed MAC protocol and report the results of our performance study.

1.5 Contributions

This work has two main contributions. The first contribution is our MAC protocol itself, including the algorithms it employs and two extensions to the RP-CDMA protocol which we propose to further improve reliability and performance. The second contribution is our performance study, in which we test and evaluate the RP-CDMA based network device and compare it to the familiar 802.11 CSMA device.

The remainder of this dissertation proceeds first with a discussion of relevant related work in Chapter 2 starting with other approaches to addressing our problem and their shortcomings, followed by a discussion of related RP-CDMA work. Chapter 3 describes the RP-CDMA Physical Layer (Phy) specifically, and we discuss how we simulate this Phy in ns-3. We describe our MAC protocol in Chapter

4, and detail our algorithms for deciding when to send packets and managing acknowledgements.

Chapter 5 opens the discussion of our performance study. In this chapter we describe our experimental setup, node configuration and method of generating data traffic across the network, as well as describe our measurements and performance parameters. Our results are presented in Chapter 6. Finally, we conclude and discuss some avenues for future work in Chapter 7.

Appendix A discusses a theoretical model of our system, which we use to verify our model on a simplified example.

Chapter 2

Related Work

In this chapter, we present a brief literature review of methods for improving wireless link reliability with a focus on their application in ad hoc networks. We begin in Section 2.1 with a discussion of some classical ways of dealing with the problem posed by interference and collisions, including carrier sense and collision avoidance (CSMA/CA), time division (TDMA), frequency division (FDMA), and code division (CDMA). After briefly evaluating these types of systems and their application in ad hoc networks, we discuss RP-CDMA as a promising choice for ad hoc networks in Section 2.2.

2.1 Other Approaches to Improving Network Reliability

We discussed in Chapter 1 the problem posed by the unreliable wireless medium, and concluded that the primary difficulty was that transmissions interfere with each other and cause packets to become unresolvable, which necessitates coordinating access to the channel. There have been many attempts to resolve this problem, with varying degrees of success [29].

An obvious way to avoid collisions is to try to avoid transmitting at the same time as other network nodes. This is the principle behind the popular carrier sense multiple access with collision avoidance (CSMA/CA) mechanism, which is probably most widely known as the basis for the MAC in the popular 802.11 series of IEEE standards [48] in which it is called the Distributed Coordination Function

(DCF). In this scheme, nodes sense the wireless medium and apply a clear channel assessment algorithm to determine whether it is occupied or not. If the channel is determined to be not busy for a short period of time then the node proceeds with its transmission. If the channel is determined to be busy then transmission is deferred until the channel is not busy, after which nodes apply a binary exponential backoff waiting period before attempting transmission again. In this way, several nodes all waiting to transmit into the channel will probabilistically have one node begin transmitting first, which will prevent the others from transmitting at the same time and thus prevent collisions. In addition to this basic CSMA/CA mechanism, 802.11 specifies an optional channel reservation mechanism in which nodes wishing to transmit will attempt to reserve the channel via special control packets that reserve the channel (RTS) and acknowledge that it is reserved (CTS). Nodes which overhear the RTS/CTS exchange will hold their own transmissions until the nodes which reserved the channel have completed their transmission.

The RTS/CTS mechanism is intended to alleviate the *hidden node* problem, which occurs when two nodes which cannot communicate with each other have a shared neighbour. In this instance both of the nodes may sense the channel to be free while the other is transmitting, which causes collisions and packet loss at their shared neighbour. The intent with the 802.11 RTS/CTS mechanism is for both sender and receiver to broadcast their channel reservation, which should prevent hidden nodes from transmitting. Unfortunately, because the communications range of the 802.11 radio is less than the interference range, there exists a space around each node in which it is impossible to detect the channel reservation RTS/CTS exchange, but nonetheless possible to cause interference at the receiver. K. Xu *et al.*[55] studied this problem and determined that an effective solution would be for nodes to artificially limit which other nodes they communicate with, which they view as a suboptimal solution to the problem. The converse of the hidden node problem is the *exposed node* problem, which is caused when a given node observes a channel reservation RTS and holds its transmission, but is actually too far away from the intended receiver to cause interference. The exposed node, therefore, wastes potential bandwidth.

Because the 802.11 CSMA/CA mechanism is vulnerable to the hidden and exposed node problems, and the RTS/CTS channel reservation mechanism is of limited effectiveness, S. Xu *et al.*[56] concluded that 802.11 is not suitable for multihop ad hoc networks. Additionally, the RTS/CTS serves as an example of the limited utility of attempting to coordinate access to the contended channel using the channel itself. If the primary problem with the shared wireless channel is packet collisions and interference, adding another layer of coordination packets into the channel may be counterproductive. This is the conclusion reached by Ray *et al.*[40], who concluded that in the ad hoc context, the addition of RTS/CTS packets eventually leads to congestion and decreased performance under increasing load.

There have been several attempts to modify the 802.11 DCF to avoid or alleviate its problems, such as adding scheduling on top of the RTS/CTS exchange [33] or coordinating non-interfering transmissions together [5], or predicting interfering transmissions using node topology [34, 19], but none of them fully overcome the limitations of the CSMA/CA and RTS/CTS approach to avoiding collisions.

Rather than attempt to avoid collisions in a single shared channel, an alternate approach is to separate transmissions into non-interfering channels. There are three common ways in which we can perform channel separation: using time, frequency or code division. A channel is therefore one of a time slot, a frequency band, or a spreading code. Each of these mechanisms has their own challenges.

In general terms, channel separation techniques in ad hoc networks typically proceed similarly, following the same basic steps:

1. Nodes are synchronized into a series of rounds. At the beginning of each round, nodes all join the control channel.

2. Nodes who wish to communicate with one another indicate their desire to do so using some channel reservation exchange.

3. Successful channel reservations either select or are allocated a channel.

4. Transmitting or receiving nodes switch to their allocated channels and exchange data.

The immediate problem with channel separation schemes that operate in this manner is the synchronization of all the nodes in order to coordinate the channel separation. Because ad hoc networks do not have any central authority or base station which can dictate synchronization information, nodes must somehow decide collectively when rounds begin and end. Fundamentally, this is the same problem that synchronization is attempting to solve: In order to not interfere with one another, nodes must coordinate their transmissions, but in order to coordinate their transmissions they must coordinate synchronization. Synchronization must be done over the wireless channel itself, which makes synchronization vulnerable to interference and packet collisions. This is precisely the problem that motivates us here to find a coordination-free ad hoc MAC.

Nonetheless, there exist channel separation mechanisms which attempt to perform each of time, frequency and code division. Zhu *et al.*[58] and Tang *et al.*[53] both attempt TDMA in ad hoc networks, with varying success, while So *et al.*[46] describe a FDMA system that envisions an 802.11 style RTS/CTS mechanism utilizing multiple frequency channels. All of these systems require very tight timing synchronization in order to operate effectively, and all are vulnerable to errors introduced by coordination failures. FDMA systems such as the one presented in [46], which have nodes switching their radios to different channels, are additionally susceptible to a novel form of the hidden node problem in which some node misses channel coordination traffic at the beginning of a round and selects the same channel as another node for transmission.

Recently, Veyseh *et al.*[54] proposed a novel FDMA system that utilizes orthogonal frequency division multiple access (OFDMA) to separate transmissions into OFDM subcarriers. OFDM is an increasingly popular choice for high capacity broadband networks [57], and has been selected by the 3rd Generation Partnership Project (3GPP) mobile phone standards group for the recently launched Long Term Evolution (LTE) standard [1]. The appealing property of OFDM based systems is that, while signals are separated into many subcarriers on separate frequencies (called tones), all communications take place in a single frequency band. The orthogonal, frequency divided subcarriers are separated at the receiver using a fast

fourier transform (FFT), which means that if several users are communicating simultaneously on different groups of subcarriers then a single receiver can successfully decode all of them simultaneously. The ad hoc protocol proposed in [54] proposes to do exactly this, and assigns users individual sets of tones which they use to communicate. The proposed protocol avoids the global synchronization of nodes by employing local synchronization between a single sending node and several receivers. The coordination problem is not completely alleviated, however, and a node which wishes to transmit must still initiate a transmission via request through a control channel and negotiate which subcarriers will be used for which transmissions. This coordination is subject to failure just as in any other protocol, and the time required to perform coordination negatively affects system performance.

Rather than coordinating access to time slots, frequency bands or subcarriers, CDMA systems coordinate access to spreading codes. The use of spreading codes in CDMA systems allows for the joint detection of overlapping transmissions, and makes CDMA systems resistant to interference and collisions. In these systems, each outgoing bit is multiplied (*spread*)[1] in the transmitter by a high frequency pseudo-noise signal, which turns the bit signal into high frequency 'noise' in the channel. At the receiver, the same code is applied to the noise, which results in the signal being recovered (*despread*). If multiple signals overlap at the receiver, the application of each signals' spreading code to the aggregate noise can return each original signal in turn, which makes CDMA very attractive as the basis for an ad hoc network radio. If we can allow for the overlapping of signals at the receiver and still recover them, then we have effectively solved the collision problem and found our reliable wireless link.

In order for a receiving node to be able to recover a CDMA signal from one of its neighbours it must first know what spreading code was used at the transmitter. This either requires nodes to have codes assigned to them ahead of time and which are globally known, or it requires nodes to coordinate their communications using a control channel. In the ad hoc context it is often impractical to assign codes to all

[1]Multiplied in this context typically means the bit signal is combined with the string of pseudo-random 0 and 1 bits using exclusive or (xor).

nodes and then distribute the book of codes to all of the other users, so codes are typically selected through an on-line coordination process as they are needed. Typical schemes of this type have been proposed by Jin *et al.*, which either have nodes elect a single node to act as a temporary or pseudo base station [21], or which rely on a channel reservation process much like RTS/CTS [22]. Thus, CDMA systems in the ad hoc context typically suffer from the coordination problem as much as their TDMA and FDMA counterparts.

However, the dominant problem in CDMA systems is not the coordination problem, but rather the *near-far* problem, which severely limits the ability of network nodes to resolve overlapping signals [38]. Without this ability to resolve overlapping transmissions there is little incentive to use CDMA at all, as the resistance to interference and collisions properties are lost. The near-far problem occurs when two incoming signals arrive with highly variable signal strengths, which is common when one transmitter is very near to the receiver and the other transmitter is relatively far away. When this happens, the far signal is not recoverable even after despreading because the near signal drowns it out. Thus, CDMA based ad hoc systems must coordinate their transmission powers as well as their codes in order to achieve high performance over irregular ad hoc topologies.

Attempts to address the near-far problem typically involve adding power control information into the channel reservation process, such as in the proposal by Su *et al.*[52], which proves to be more effective than similar schemes which do not include power control. But by this point we are quite far from our intended aim, which was to find a coordination-free ad hoc MAC protocol. If we wish to use conventional CDMA detectors in our ad hoc network we must coordinate the distribution of spreading codes and the transmission powers of each node in the network, all in a distributed fashion. Even with all of this, coordination failures can still lead to reliability problems, as all of this coordination is done over the wireless channel itself.

In summary, single channel ad hoc systems tend to suffer from packet collisions, and attempts to avoid collisions using CSMA/CA are prone to error. Attempts to resolve these errors with coordination functions such as the 802.11 RTS/CTS mech-

anism are not entirely effective, and may actually be counterproductive. Thus, it is natural to investigate channel separation schemes for ad hoc networks as a means to separate different users' transmissions from one another. Channel separation systems typically involve either tight time synchronization or some other form of distributed coordination to control access to a limited number of time slots or frequency bands, which reduces the error potential from all traffic in the single channel to only the coordination traffic in the coordination channel, but which nonetheless is still prone to error in the same way. CDMA systems are promising due to their inherent resistance to interference, which promises to remove the requirement for node synchronization, but they also introduce new problems of code and power coordination between nodes.

It is with this in mind that we turn our attention to Random Packet CDMA (RP-CDMA), which uses a novel packet format to overcome the requirement to coordinate spreading codes, and a powerful multiuser detector to overcome the near-far problem. With these problems resolved, RP-CDMA appears to be a promising candidate for a reliable ad hoc wireless link.

2.2 Random Packet CDMA

RP-CDMA was first proposed by Kota and Schlegel *et al.* [28, 42], as a means of uncoordinated, random channel access. In this initial effort, data packets were transmitted in two parts by sending the packet header over a common header channel and the packet payload in a randomly selected data payload channel. The header portion of each packet contained only enough information to identify and decode the remainder of the packet in the payload channel, and the header channel was thus comparatively lightly loaded under the assumption of large payloads. Each node would listen to the header channel in order to identify transmissions in the payload channels, and thus no coordination between nodes was required in order to exchange spreading codes. Packet payloads were decoded in a *multiuser detector*, which is a CDMA receiver capable up decoding up to K overlapping transmissions simultaneously. The header channel was modelled as a lightly loaded Spread

ALOHA channel, and the system was found to be limited by the capability of the multiuser detector, K. With the use of iterative decoding in the multiuser detector, the authors concluded that the system could approach the Shannon limit of the multiple access channel and was therefore limited by the capability of the multiuser detector, as opposed to collision-limited in the common header channel. This is an important result, since the common header channel is the only contended resource in the RP-CDMA system. By showing that random access to this common resource was sufficient to approach the information-theoretic bound on capacity, the system was freed from the requirement to coordinate between nodes for wireless medium access. This work was confirmed with traffic scenarios involving alternating light and heavy users, along with a capacity analysis in Kempter *et al.*[24].

Kempter [25] subsequently examined the application of various multiuser detectors in the RP-CDMA system in an effort to determine what kind of multiuser detector would be required to match the initial results presented in [42], which assumed an ideal detector capable of decoding any packet identified from the header channel. This work confirmed that the header channel is not collision-limited, but rather limited by the capability of the receiver in both the header detector and payload multiuser detector. Various types of multiuser detectors were considered in both [25] and [26], which concluded that a Partitioned Spreading CDMA receiver performed best both in terms of reasonable spreading code lengths and resistance to the near-far effect, but critically illustrated that high system performance could be maintained even under high load [26].

Nagaraj *et al.* applied a CSMA channel access scheme to the common RP-CDMA header channel in [35], and determined that overall system throughput could be improved over the spread ALOHA random access case under particular traffic scenarios. Subsequently, Nagaraj *et al.*[36] reexamined the performance characteristics of a random channel access scheme with a multiuser detector, and concluded that as multiuser detector capability goes to infinity, system throughput asymptotically approaches the optimal value. This result is both derived analytically and confirmed with simulations on a fully connected network with a simplified packet reception model.

Although random access may be optimal in the limit of multiuser detector capability, it is not optimal with limited detector capability. For the case of the limited multiuser detector, Ghanbarinejad *et al.* proposed an adaptive probabilistic MAC protocol for multiuser detector capable systems in which nodes used channel traffic estimates to feed a probabilistic model of whether a node transmitted or not in a given time slot [16, 17]. This work is similar in spirit to that presented by Kempter *et al.*[24], which also adopted a feedback based adaptive MAC, and also concluded that an adaptive probabilistic model is suitable for maintaining traffic loads below the multiuser detector limit, K, compared to ALOHA style random transmission.

We found that much of the literature is focused on the relative performance of various types of multiuser detectors, such as Kempter *et al.*[25, 26], or interested in developing MAC protocols which maximize transmitter parallelism, such as Ghanbarinejad *et al.*[16]. As a result, network configurations or simulations are typically simplified to include a single receiver and several transmitters, as in [16], or simply to consider the probability of header collisions in the common header channel as a measure of successful transmission, as in [42]. These approaches address the transmission side of the packet transfer process, but ignore the state of the receiving node and specifically ignore whether the intended receiving node is actually prepared to receive when a transmission begins. The CSMA work of Nagaraj *et al.*[35] includes a notion of node behaviour during neighbour transmissions in their performance evaluation, but this work appears to also consider only the probability of a packet header collision and the probability of exceeding the detector capability as the basis for system throughput. While these performance measures may be appropriate in the case of a base station style of network in which multiple nodes all communicate only with a powerful base station, it is not clear that it applies to the multihop ad hoc network case. Specifically, in an ad hoc network it is not only required that there be no header collision for a transmission to be successful, but it is also required that the intended receiver *not be transmitting* at the same time, as a half duplex radio cannot simultaneously transmit and receive. This aspect appears to often be overlooked in the literature we surveyed. A node may find the header channel unoccupied when it wishes to transmit, but if it is currently receiving one

or more packet payloads - which may or may not be intended for it - then it cannot switch to transmit unless it is willing to drop the current incoming packet(s). Similarly, a node may find the header channel unoccupied when having a packet to send to some neighbouring node, but if the neighbour is transmitting a packet payload then it makes no sense to begin transmission to it until it finishes transmitting itself. Thus, the state of the receiving node is an important factor when deciding whether or not a particular transmission was successful.

With this in mind, we are focused on the application of RP-CDMA in a multihop ad hoc network. Our survey of the literature provides evidence that the application of a multiuser detector in ad hoc networks may have significant reliability and performance advantages. Specifically, we believe that a sufficiently powerful multiuser detector can let us approach the capacity of the multiple access channel, but we feel that realistic ad hoc network conditions have not been adequately considered to this point. As such, we seek to confirm that RP-CDMA can be applied successfully in a general multihop ad hoc environment via network simulation.

Chapter 3

Random Packet CDMA

RP-CDMA is a novel wireless CDMA protocol in which each packet is encoded in two parts [42]. The packet header is encoded with a common spreading code that is known to all nodes, and the payload is encoded with a randomly generated spreading code. An identifier for the payload spreading code is placed in the header, so, as a receiving node decodes the packet header, they obtain the code used to decode the payload. Thus, each packet is self contained and any network node is able to decode any packet in the network without any requirement to know beforehand which payload code was used. When combined with a multiuser detector, RP-CDMA enables a single node to receive several packets concurrently.

In this chapter, we first discuss RP-CDMA in general and follow with the details of our simulated implementation. Our general discussion of RP-CDMA includes its fundamental characteristics, including packet composition and transmission, spreading code selection, effective bandwidth and channel capacity. We then discuss multiple packet reception and how the multiuser detector enables the reception of concurrent packets. These general properties are put into the context of our reliable link properties from Chapter 1, and we will see how RP-CDMA addresses each of them. With this theoretical discussion completed, we describe in detail how our simulated RP-CDMA network device works, and describe the algorithms used in our simulated device in order to transmit and receive packets.

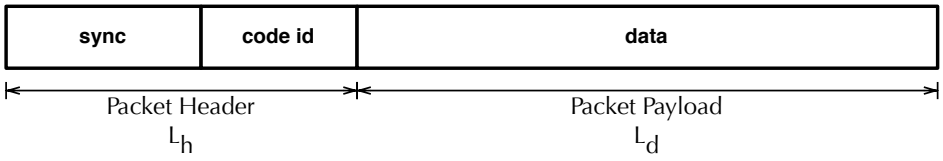


Figure 3.1: RP-CDMA packet structure from Schlegel *et al.*[42]

3.1 Fundamental Characteristics

As a CDMA spread spectrum system, RP-CDMA is inherently resistant to interference and noise in the channel, just like other CDMA systems. It is this property of spread spectrum that makes it a popular choice for wireless communication systems, and which underlies the requirement to employ spread spectrum in the ISM frequency bands which are increasingly used for wireless systems of all types [12].

The novel property of RP-CDMA that distinguishes it from other CDMA protocols is the way in which spreading codes are used to separate packets into individual channels without any requirement for a packet sender and receiver to coordinate channel selection or access before data transmission. This distinguishes it from the bulk of other CDMA protocols, which almost invariably either assign each node a particular single spreading code to use for all transmissions [21], or which require nodes to agree beforehand on which code will be used for a given transmission using some kind of channel reservation system or through a coordination channel [22]. We saw in Chapter 2 that these kinds of systems tend to suffer from problems with the requirement for nodes to coordinate with each other before communicating, which fundamentally limits their effectiveness.

It is the RP-CDMA packet structure which enables asynchronous, uncoordinated channel access by transmitting nodes. This structure, as described by Schlegel *et al.*[42], is shown in Figure 3.1. In this figure, we can see that the packet is divided into two sections, the packet header and the packet payload. The header consists only of the synchronization bits (or access preamble), and the code id, which indicates the spreading code used to encode the packet payload and any other information needed to decode the payload. The receiver similarly consists of two stages: the header detection first stage, and the multiuser detector second stage. In

the first stage, the receiver synchronizes to the header synchronization bits and acquires the timing of the incoming packet. Once timing is recovered, the first stage of the receiver decodes the code id portion of the packet, which contains all of the information required to decode the packet payload, and hands this off to the second stage along with the timing information. The second stage then decodes the packet payload, recovering the data.

Because all nodes use the same spreading code to encode packet headers, any node in the network can decode any packet header, and therefore can decode any packet in the network. Fundamentally, this enables network nodes to operate asynchronously, as there is no requirement to coordinate transmissions with other nodes in order to exchange spreading codes or timing information.

3.1.1 Spreading Code Selection

The way in which the packet header is transmitted, in terms of spreading code used or data encoding scheme, does not have to be the same as that of the payload in the RP-CDMA system. It is in fact reasonable to select a stronger spreading code or higher transmission power for the packet header than for the payloads because the header channel is the only practically contended channel in the RP-CDMA system [25].[1] Thus, it is possible that multiple headers using the same spreading code could overlap at a single receiver, and because the header detection stage is critical in acquiring the packet timing, we may therefore employ a longer spreading code for the packet header in order to achieve more gain, transmit headers with more power, or we may choose a spreading code that has a low degree of autocorrelation.

For the payload spreading codes, we may employ any class of spreading codes that we like, so long as there are sufficient codes such that it is unlikely that any two neighbouring nodes will select the same code at the same time. Schlegel *et al.*[42] suggest maximal length sequences (m-sequences) as possible candidates due to their desirable autocorrelation and pseudorandom properties, but settle on

[1] Technically, the payload channels are also contended, but we shall see that by allowing for a very large number of payload channels the probability that two neighbouring nodes simultaneously select the same one is insignificant, which is why we say the header channel is the only *practically* contended resource.

random codes for the purposes of their analysis. This is a reasonable choice in the asynchronous ad hoc environment, since asynchronously arriving packets will be randomly offset from one another anyway, which will change the correlation values between any specifically chosen set of codes.

3.1.2 Bandwidth and Channel Capacity

The bandwidth of a CDMA system is given by the length of the spreading code employed and the bandwidth of the unspread signal, using $W_s = GW_d$, where W_s is the bandwidth of the spread signal, G is the spreading gain (which is the same as the length of the spreading code), and W_d is the bandwidth of the original data signal, which depends on the rate of carrier modulation, which for binary signalling is simply $W_d = 2R$, where R is the data rate [51]. RP-CDMA is no different from other CDMA systems in this regard, and the bandwidth occupied by RP-CDMA depends on the length of spreading codes used in both the packet header and payload. Because RP-CDMA is intended to be used for overlapping signals, the codes chosen should be long enough to provide a low error rate in the low signal to noise environment. Throughout our work, we assume a data rate of $R = 1$ Mbps, and thus we will always assume the bandwidth of the RP-CDMA signal is given by $W_s = GW_d = 2KR = 2K$, where we assume that the number of users supported by the multiuser detector, K, is equal to the length of the spreading codes used, G, so we assume $K = G$. It is worth pointing out that $W_s = 2K$ is a somewhat pessimistic assumption, as more sophisticated signalling can achieve modulation rates less than the data rate, and the number of users supportable in some multiuser detectors is greater than the spreading code length. In particular, it should be possible to achieve $K = 2G$ with sufficiently powerful decoders [42], but we assume $K = G$ throughout this dissertation.

The upper bound on the capacity of the channel is given by the Shannon limit [51]:

$$C = W_s \log_2(1 + SNIR) \tag{3.1}$$

Where C is the channel capacity in bits per second, W_s is the bandwidth of the

channel in Hertz, and SNIR is the signal to interference plus noise ratio, $SNIR = \frac{P_s}{N_0+\sum_{i=1}^{N} P_i}$. SNIR is simply the ratio of the signal power to the cumulative other noise in the channel, which is just the thermal noise, N_0, plus the sum of the signal strength of all the other transmissions in the channel, denoted by the sum of the individual signal strengths, P_i.

In RP-CDMA , the upper limit of the channel capacity thus depends on the bandwidth used - which itself depends on the spreading code length - and the relative signal strengths of all the signals in the channel. For our purposes, we are mostly interested in the performance of our MAC protocol on top of the RP-CDMA link. It has been established in the literature that with a sufficiently capable multiuser detector it is possible to approach the Shannon limit for channel capacity [42, 25, 43], and so we work within this result. Specifically, we assume a fixed data rate for our transmissions of 1 Mbps, and leave the details of the data encoding, transmission power levels, and multiuser detector decoding processes alone. This is not unusual, as the literature routinely assumes, for the purposes of analysis, that a multiuser detector of capability K can successfully decode up to K overlapping signals without error, leaving aside the same transceiver operation details, as in [16, 36].

3.2 Multiple Packet Reception

A basic iterative multiuser detector consists of a simple interference suppression stage in front of a parallel bank of single channel decoders [44, 43]. The interference suppression stage takes the aggregate received signal in and passes it to the single channel decoders. As the decoders successfully decode their data, they pass it back to the interference suppression stage, which then modifies the received signal passed back to the decoders. This feedback loop between the decoders and the interference suppression stage allows for the iterative decoding of all of the individual signals comprising the aggregate received signal. A multiuser detector with capability K can decode up to K overlapping signals simultaneously, with the decoders made up of a fixed bank of decoding circuits, or possibly software processes started up on demand by the receiver as packets arrive [42, 28].

There have been several types of multiuser detector proposed and studied in the literature, such as the simple linear matched filter, the decorrelator, and the minimum mean squared error (MMSE) decoder [44]. There are also decoders which perform iterative signal cancellation, such as that proposed by Holtzman [18], and which integrate signal cancellation with other techniques, such as in Partitioned Spreading [41, 25]. Iterative cancellation decoders perform just as well as other decoders when the signals arrive with equal power, but have an advantage when multiple signals arrive with different power levels [42, 8, 25]. In this case, the iterative cancellation of signals allows the receiver to successfully decode all of the incoming packets from strongest signal to weakest, in what is sometimes called 'onion-peeling' [44, 25]. This property provides the receiver with inherent near-far resistance [41], which we have already seen is a problem in CDMA systems, and particularly in ad hoc CDMA systems.

Because of their performance with unequal signal powers, we will assume throughout this dissertation that the multiuser detector employed in the RP-CDMA device is of the iterative signal cancellation kind. The use of an iterative cancellation multiuser detector in RP-CDMA enables the final desirable property of the system by eliminating the requirement for nodes to coordinate their transmission power levels, which is the last problematic coordination point for CDMA systems. The employment of an iterative cancellation receiver allows RP-CDMA network nodes to transmit with whatever power level seems appropriate for their purpose, and specifically without input from their neighbours.

In addition to eliminating the need for nodes to coordinate with one another before transmitting, the capability of the multiuser detector to better resolve signals when they are distributed in signal strength indicates that we can utilize transmission power as a second axis - in addition to time - in which to separate transmissions within a given frequency band. Signals arriving in the same frequency band at the same time can be successfully recovered using iterative cancellation from the most powerful signal to the weakest signal. This property turns the near-far problem from a weakness of CDMA systems to a strength. We will return to this in Chapter 4.

3.3 Desirable Properties

In summary, because RP-CDMA employs code division, individual signals are resistant to external interference in the same way as traditional CDMA systems. By placing payload spreading codes in packet headers, each packet is transmitted in a private channel and is decodable by any node in the network, with no requirement for coordination between nodes before transmitting data. Finally, the ability to simultaneously decode multiple packets spread over a range of signal strengths eliminates the need for nodes to coordinate their transmission powers. Thus, there is no requirement for nodes to coordinate their transmission at all in the RP-CDMA system, and, in these ways, RP-CDMA can achieve much of what we want from a wireless link:

1. Random payload codes provide packet level private channels, except for packet headers.

2. Code division provides resistance to interference, both external and from other packets.

3. Multiple packet reception and packet formatting eliminate the need for channel access coordination.

Having now discussed what RP-CDMA is, how it works, and why it seems promising as a basis for our reliable link, we can now move on to a discussion of how we simulate it in ns-3.

3.4 Simulation of RP-CDMA

We simulate RP-CDMA in ns-3 through a custom network device. Since our aim is to design a MAC protocol that uses a RP-CDMA based Phy to build an efficient wireless network device, we must simulate the RP-CDMA Phy. We simulate this down to the level of packets being sent through a wireless channel to other nodes in the network. At this level we simply pass packets between nodes as a series of bytes, and so do not simulate the actual encoding of bits or transmission waveforms.

In general, a packet passed from the MAC layer to our RP-CDMA based Phy will have a randomly selected payload spreading code assigned to it, the RP-CDMA Phy header prepended, and be sent into a common wireless channel from which it is received by all other nodes in the channel. The receiving Phy must decide whether the received signal can be decoded based on the state of the receiver and the instantaneous channel noise conditions, and then perform packet decoding in the multiuser detector. In order for packet decoding to happen successfully, the Phy must therefore simulate the working of a multiuser detector, as well as track the noise level in the channel in order to measure interference.

In this section, we describe our simulated RP-CDMA Phy from the point of view of a packet being sent down from the MAC layer from a transmitting node. We follow the packet as it passes through the channel and into a receiving Phy. In so doing, we describe our RP-CDMA Phy header, how we track channel noise, decode packet headers and payloads, and finally simulate the working of the multiuser detector and pass packets up to the receiving MAC layer.

3.4.1 Simulator Preliminaries

Because we will discuss our algorithms, experimental design, and results in the context of the simulator, it is useful at this point to briefly describe the simulator we use, how it functions, and to what level of detail we simulate the network.

All of our simulations are performed with with the ns-3 network simulator.[2] Though similarly named, ns-3 is not derived from the more familiar ns-2 project[3], but is an entirely new codebase and the respective development teams are not affiliated. That said, ns-3 aims to provide a highly realistic discrete event driven network simulation stack upon which to perform network research down to the packet level [30]. To this end, ns-3 provides a complete network stack, which allows researchers to create realistic simulations from the application layer down to the physical layer and into physical channels. At the application layer, programs have a normal network sockets interface which takes packets and bytestreams. At the physical layer,

[2] `http://www.nsnam.org/`

[3] `http://www.isi.edu/nsnam/ns/`

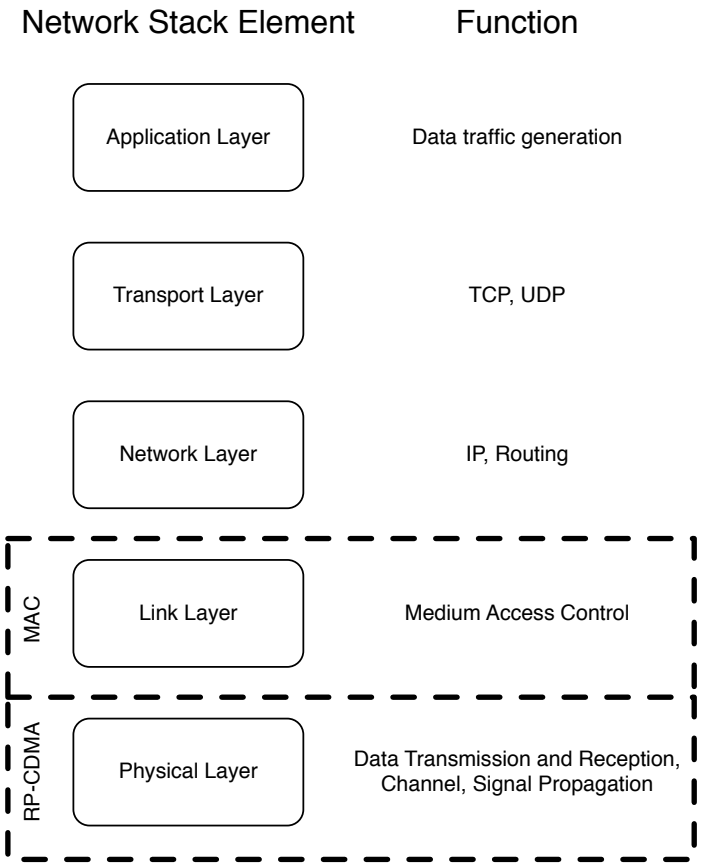


Figure 3.2: The ns-3 network stack. We created a network device with Link and Physical layers, indicated with the dashed lines. The RP-CDMA transceiver works at the Physical layer, and our MAC protocol works at the Link layer.

channels provide facilities to model signal fading, interference and propagation delay. The networking stack itself is modelled on the real world Linux kernel network stack, to the degree that the actual network stack from a host Linux system can be substituted for the simulator stack should a researcher wish to do so. Since the simulator is entirely software, it is possible for a researcher to simply modify various parts of the realistic network stack and run simulations to evaluate performance. This is, in fact, the network simulator's *raison d'être*.

The ns-3 network simulator is an event driven simulator. This means that the passage of time is simulated simply by executing a series of scheduled events. Any component in the simulation can schedule an event to occur at some point in the future and defines what action to take when the event is triggered. Once triggered, the simulator performs the scheduled action. In this way, we can incorporate the passage of time into our simulations. For example, when a packet is transmitted by some node into the channel, we calculate the arrival time of that packet at each other node in the channel by considering the transmission propagation speed and the distance between the nodes. Once the arrival time is calculated, we can schedule the simulator to process the arrival of the packet at that time. In this way, the simulator can drive the simulation simply by processing events in the order of their scheduled execution times.

In this context, our work involves the simulation of a MAC protocol built upon a RP-CDMA based transceiver. As this type of device does not already exist in the simulator codebase, we have therefore created from scratch a new ns-3 network device which includes a Link and Physical layer. Figure 3.2 illustrates where in the ns-3 network stack our device resides, which we have outlined with a dashed line. At the Link layer we implement our MAC protocol, which decides when to transmit and when to receive, and handles our link layer acknowledgements. Below the MAC is the RP-CDMA based Phy, which handles the transmission of packets into the wireless channel and packet reception. The remainder of this chapter describes the simulation details of our Phy, and Chapter 4 describes the working of our MAC protocol.

3.4.2 Header Description

In the literature, the RP-CDMA header is usually described as containing only the synchronization bits (or access preamble) and the payload code id [42, 28, 25].[4] In theory, these two fields are all that is minimally required in order to identify the packet payload and decode it, as they are sufficient for the radio receiver to first lock on to the incoming transmission using the synchronization bits, and then identify the payload spreading code to hand off to the multiuser detector for decoding. In practice, the receiver needs to know other bits of information about the packet in order to complete the decoding process, such as the length of the packet, any error correction information, and any particulars about the encoding process such as the amount of redundant information or information rate. This payload information could be included in the packet payload itself, but we would then be putting the payload decoder in the position of needing to decode some of the payload in order to get all of the information required to decode the payload.

In our implementation, we include the synchronization bits, payload spreading code id, packet length, packet cyclic redundancy check (CRC) and the packet encoding information rate in the RP-CDMA Phy header. The extra fields beyond the synchronization bits and the payload spreading code are included with the Phy header instead of with the packet payload because they enable the multiuser detector to perform decoding of the payload without any additional information. Our aim in doing this is to simplify the decoding of packet payloads in the multiuser detector. To this end, we include the packet length so the multiuser detector will know how long a packet is going to be, and thus when a specific transmission is expected to end. The CRC is included so that the multiuser detector can easily perform basic error detection over the entire packet, and the information rate informs the multiuser detector how much of the payload is redundant information to be used for error detection and correction. Together, these fields enable the multiuser detector to decode the packet payload without having to consult any information inside

[4] We assume that the authors mean 'code id' to be any information required to decode the packet payload and not only the actual payload spreading code, but we could not find a detailed header specification in the literature.

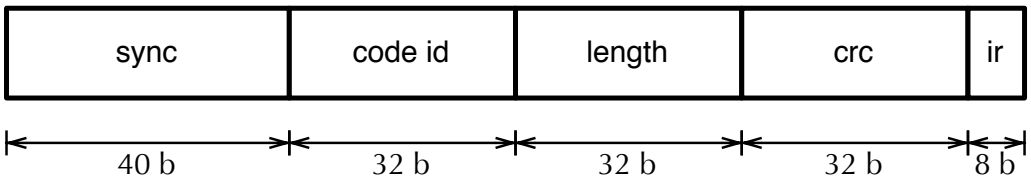


Figure 3.3: RP-CDMA Phy header, showing header fields and their lengths.

the payload itself. Our Phy header showing each of these fields and their respective lengths in bits is depicted in Figure 3.3.

For the sync portion of the header we use 40 bits. This is slightly shorter than the 802.11 Short Physical Layer Convergence Procedure (PLCP) Preamble of 72 bits [48], but on the same order as the more recent 802.15.4 standard which also uses 40 bit preambles in its Binary Phase Shift Keying (BPSK) mode [47]. The number of bits in the sync header is therefore reasonable, and would be feasible in a physical implementation. We assign 32 bits for the payload spreading code identifier, which gives us a possible 2^{32} codes to choose from when randomly selecting a spreading code for a packet. We allocate 32 bits for the packet length, which allows us to specify packet lengths up to 2^{32} in size - much larger than we envision being required, and also more than the 16 bits allocated for the same purpose by 802.11 [48]. We allocate 32 bits for the CRC, which allows us to make use of popular 32 bit CRC polynomials such as that specified by Ethernet [50], and which compares favourably to the 16 bit CRC checks in both 802.11 and 802.15.4 [47, 48]. Lastly we allow 8 bits for the information ratio, which allows us to specify up to 2^8 values for the level of redundancy in the message, or the information rate, which is more than required to specify typical values in the range of 2-12, or unusual values up to 256 [44].

3.4.3 Packet Transmission

A packet is passed to the Phy layer for transmission with or without a payload spreading code assigned. This allows the MAC to optionally specify which payload spreading code to use for a transmission when sending either acknowledgement packets or retransmitting a previously sent packet. If no payload code is specified

by the MAC, then the Phy will randomly select a spreading code. With the payload spreading code decided, the Phy will construct the Phy header for the packet, as described above in Section 3.4.2, and begin the process for transmitting a packet into the channel. Once the packet has been transmitted, the Phy will return the payload spreading code to the MAC.

At any time the Phy can be in one of four states:

1. The Phy can be idle (IDLE), and is neither transmitting a packet nor receiving a packet

2. The Phy can be receiving either a packet header (RX_HEADER) or a packet payload (RX_PAYLOAD)

3. The Phy can be transmitting a packet payload (TX_PAYLOAD)

4. The Phy can be transmitting a packet header (TX_HEADER)

In our system the MAC is aware of the Phy state, and makes the decision of when to pass packets down to the Phy based on its own assessment of when the device should transmit. The Phy is responsible only for transmitting the packets passed to it from the MAC, and the MAC only passes packets to the Phy for transmission when the Phy is in a state such that it can transmit them.

We can enumerate the behaviour of the Phy when a packet arrives from the MAC for each of the possible states listed above:

1. If a packet arrives from the MAC while the Phy is idle, then the Phy will transmit the packet header and then the packet payload normally. The Phy state will go from IDLE to TX_HEADER, and then to TX_PAYLOAD as the packet header and payload are transmitted respectively.

2. If the MAC passes a packet to the Phy while the Phy is receiving one or more packets and is in state RX_HEADER or RX_PAYLOAD, all packet receptions will be cancelled and the Phy will switch to the transmit state in order to transmit the packet header and payload.

3. If a packet arrives from the MAC while the Phy is transmitting a packet payload, then the Phy will transmit the packet header and payload concurrently with the already outgoing packet payload, with the Phy state switching to TX_HEADER and then back to TX_PAYLOAD as the header and payload go out. This is the basis of Simultaneous Transmission, which we discuss in Section 4.2. If a packet arrives which brings the number of simultaneous transmissions up to the limit of the multiuser detector, K, then the Phy signals the MAC that no more packets can be transmitted.

4. The MAC will not pass a packet to the Phy when it is already transmitting another packet header and is in the state TX_HEADER.

We see then that once a packet is received from the MAC, the Phy will begin transmitting it into the channel.

3.4.4 Channel Characteristics

Once transmitted from the Phy into the channel, the channel schedules packet arrival at all other nodes in the channel. The distance, d, between the transmitting node and each other node is calculated, and the transmission power, P_{tx}, is attenuated using the ns-3 Log Distance Propagation Loss Model, which decreases the power at the receiver, P_{rx}, using the formula:

$$P_{rx} = P_{tx} - L_{ref} - 10 \cdot 3 \cdot \log_{10}(d/d_{ref}) \tag{3.2}$$

Where L_{ref} is the loss at the reference distance, d_{ref}, all distances are in metres (m), and power is in decibels referenced to one milliwatt (dBm). For distances less than the reference distance, $d < d_{ref}$, the signal strength is not attenuated, and so $P_{rx} = P_{tx}$. In ns-3, the default reference distance is 1.0 m, with $L_{ref} = 46.6777$ dB, which is the Friis loss at 1 m and 5.15 GHz [30].

Once the power is attenuated with the propagation model, the packet is scheduled to arrive from the transmitter at each other node in the network using a straightforward constant speed delay, where the propagation delay is given by:

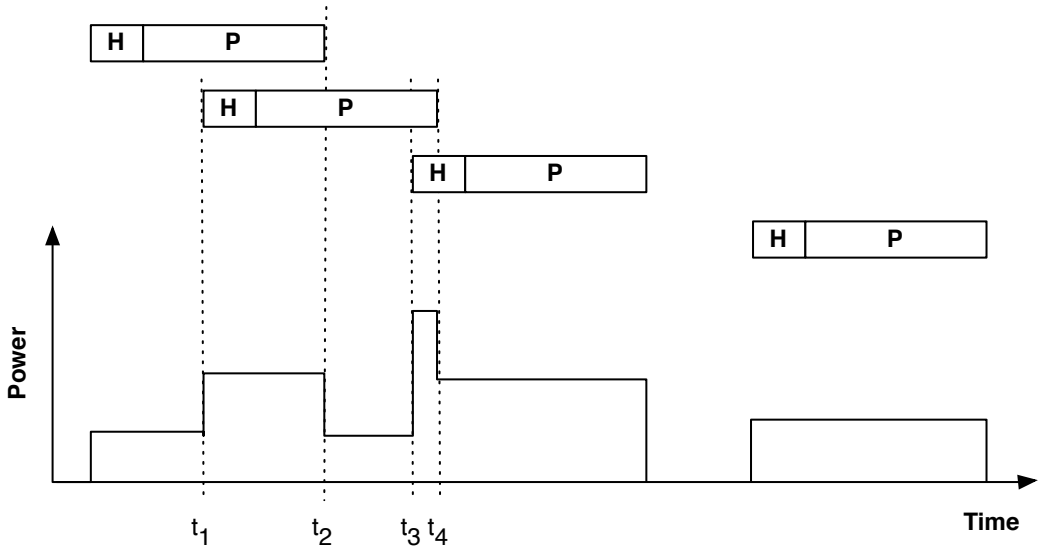


Figure 3.4: Interference Model. We track the noise in the channel by packets.

$$T_{delay} = d/c \tag{3.3}$$

Where $c = 299792458$ m/s, the speed of light.

3.4.5 Interference Tracking

Whenever a packet arrives at a receiving Phy from the channel, the signal strength is added to the interference model object, which records the start and end times of each signal, along with the received signal strength in dBm. The interference model therefore maintains a continuous trace of the cumulative signal strength measured by the receiver, as illustrated in Figure 3.4.

Using this trace it is possible for the Phy to determine the signal to noise ratio in any time interval. Given a start and end time for a signal, the interference model can provide the noise level in the channel for each chunk of time in which the noise level is constant. For example, in Figure 3.4, the noise level present in the channel between the times t_1 and t_4 - during which the second packet is received - is broken down into three noise level chunks, covering the time periods t_1 to t_2, t_2 to t_3 and t_3 to t_4. For each of these periods, the signal to noise ratio can be calculated for the second packet, from which we can calculate the probability of a bit error in the

decoding of the given signal, and therefore calculate the probability of an error in decoding the packet.

3.4.6 Packet Reception

Upon arriving at a receiver the packet signal strength and duration are recorded in the device interference model, as described in Section 3.4.5. Once the interference from the transmission is recorded, the Phy decides if it can actually decode the packet. This happens in two phases: header reception and payload reception.

Because the packet header is sent using the common header spreading code, the Phy is capable of decoding only one header at a time. The Phy must successfully decode the packet header, extract its contents as described in Section 3.4.2, and then pass off the payload reception job to the multiuser detector, which then decodes the packet payload.

When a packet arrives from the channel at the receiving Phy, the Phy first decides if it will be able to synchronize to and decode the header portion of the packet. In general, the Phy will be able to decode the header if the signal power at the receiver is greater than some minimum detection threshold, and the device is in a state such that it will be capable of detecting and devoting resources to header and payload decoding. Therefore, when a packet header arrives at the receiving Phy, the Phy will accept the packet and schedule the decoding of the packet header unless any of the following criteria is met:

1. The signal strength is below the device detection threshold.
2. The device is transmitting.
3. The device is receiving another packet header (this is a header collision).
4. The device is already receiving K packets.

If the packet does not meet any of these criteria, then the Phy will accept the packet and schedule a header decoding event for the time when the header will have completely arrived at the receiver. Once the header has arrived, the Phy will calculate the probability of successful header detection based on the signal to noise ratio

in the interval between when the header began and ended, and compare this probability to a uniform random number between 0 and 1 to determine header reception success. If successful the Phy will decode the header and then pass the packet information, including the payload spreading code and packet length, to the multiuser detector for payload decoding.

3.4.7 Multiple Packet Reception

The multiuser detector can decode up to K packets simultaneously. In our implementation, the multiuser detector simply schedules a packet decoding event when the packet is due to have finished arriving, and then computes the probability of successful packet reception based on the signal to noise ratio measured by the interference model in the time between when the packet payload started and ended at the receiver. This probability of successful reception depends on the characteristics of the receiver and the type of error correction coding used, which we are free to substitute into our model based on the type of multiuser detector we wish to test in our simulator. In particular, for the ideal receiver which always successfully decodes K packets regardless of interference, we have the probability of successful reception equal to 1. So in the ideal case, as long as the packet is not dropped by the Phy for one of the reasons above, then it will be successfully received by the multiuser detector. In this work, we performed all of our experiments with ideal receivers.

Once decoded by the multiuser detector, the received packet is passed up to the MAC layer for handling and processing.

3.5 Summary

In this chapter, we began by discussing the properties of RP-CDMA as proposed by Schlegel *et al.*[42]. This discussion included the fundamental properties of RP-CDMA , including packet format, selection of spreading codes, required bandwidth, and the properties of the multiuser detector. We related these properties to our desired link properties from Chapter 1, and concluded that RP-CDMA could offer us private channels for each packet, interference resistance, and coordination free

transmissions, and was thus a suitable candidate for a reliable wireless ad hoc link.

With the theoretical evaluation complete, we began the discussion of how we simulated RP-CDMA in the ns-3 network simulator. We began with the basic details of how the simulator works and then described our RP-CDMA packet header format, which is designed to allow a packet payload to be decoded by the multiuser detector without having to refer to any information in the packet payload itself. We then traced a packet through the RP-CDMA Phy, as it was passed down from the MAC, through the Phy, and into the channel. Once in the channel, we described how a packet arrives and is processed by the receiving node Phy, including interference tracking, header collision detection, and multiple packet reception in the multiuser detector, before being passed up to the receiving node MAC.

Having completed this discussion of RP-CDMA and how we simulate it, we can now move on to the discussion our MAC, which drives the RP-CDMA Phy and is one of the main contributions of this work.

Chapter 4

MAC Description

In this chapter we will describe the details of our MAC protocol, which we designed to operate on top of the RP-CDMA Phy. This MAC protocol and the accompanying performance study is the main contribution of our work, and our aim is to demonstrate that the RP-CDMA based network device can achieve relatively high reliability and performance on an ad hoc network with a simple MAC protocol such as the one described in this chapter.

Before describing the details of our MAC protocol, we first motivate its design in Section 4.1. We will see that the addition of the multiuser detector changes the nature of the link reliability and performance problem, which motivates us to propose two extensions to the standard RP-CDMA protocol: Simultaneous Transmission and Payload Channel Acknowledgements. These extensions are designed to make maximum use of the multiuser detector, improve system performance, and reduce congestion in the common header channel. After describing these extensions in Sections 4.2 and 4.3, we then describe our MAC protocol in Section 4.4, and discuss acknowledgement policies in Section 4.5.

4.1 Motivation

Our investigation into the application of RP-CDMA in the multihop ad hoc context was initially motivated by the observation that modern wireless ad hoc systems suffered from the problems of packet collisions and interference, which fundamentally limited their reliability and performance. Initially, we experimented with RP-

CDMA by modifying the ns-3 802.11 stack to simulate a multiuser detector at the physical layer, leaving the 802.11 MAC intact. These early experiments found that packet losses due to collisions were indeed significantly reduced, as we had hoped, but that system performance compared to the 802.11 CSMA model, while better, was not dramatically improved. Based on these early experiments, we decided to abandon the 802.11 model entirely and implemented RP-CDMA in its own simulated device from scratch. Our early experiments with this device found, again, that even with our own MAC on top of the multiuser detector, our performance was improved over the previous result, but not dramatically so.

When we investigated what was limiting the performance of our ad hoc network even after the elimination of losses due to interference, the reduction of packet collisions to negligible levels, and the removal of any need for nodes to coordinate their transmissions, we found that there were two main causes of performance degradation: First, the dominant cause of packet loss was due to the receiving node being in the transmit state when a packet arrived, and so the half duplex nature of the radio became a significant problem, since nodes cannot both transmit and receive simultaneously.[1] Second, because an ad hoc network node may transmit to any one of its neighbours either for packet forwarding or to actually deliver a message, nodes must listen to their neighbours for incoming packets. This requirement fundamentally limited the performance of the system, as nodes had to wait for their neighbours to finish transmitting a packet before commencing their own transmissions.

Coupled with our observation that nodes were spending significant time waiting for neighbour transmissions to finish, we observed that the multiuser detectors were not being significantly loaded at any time, despite each node having large queues of packets waiting to be transmitted. Rather, we found that with only one or two nodes in a group transmitting at any given time (and the rest all waiting for the transmissions to finish), the multiuser detectors in each node did not typically have to handle more than one or two incoming packets at a time. This is a problem some-

[1] Notwithstanding the recent work by Choi *et al.*[10], who have constructed a prototype full duplex radio using a clever form of signal cancellation.

what unique to an ad hoc system where nodes must all listen to their neighbours, and absent from a base station type system where all nodes transmit to the base station. In a base station style system, such as that examined by Ghanbarinejad *et al.*[16], maximizing the number of concurrently transmitting nodes is a viable strategy to make the most use of multiuser detectors. In contrast, maximizing the number of concurrently transmitting nodes in an ad hoc system only increases the probability that the intended recipient of any transmission is transmitting itself, and that the packet will be lost due to the half duplex nature of the radio.

Our early experiments with RP-CDMA in ad hoc networks were therefore of limited success. Link reliability was significantly improved, but as we increased the system load we still lost packets due to the half duplex nature of the radio, and system throughput was limited by the need for nodes to take turns transmitting to each other. Our problem therefore became how to limit the time nodes spent transmitting while simultaneously increasing the number of packets transmitted.

Our solution to this problem lies in the multiuser detector. Our observation that nodes' multiuser detectors were relatively under-utilized while the transmit queues filled up illustrated that our system was no longer constrained by the receiver capability, as it was when the system was limited by packet collisions and coordination, but rather the system was now transmit capability limited. This observation has motivated the design of our MAC protocol, in which we have coupled the powerful multiuser detector with an aggressive transmission mechanism designed to minimize the amount of time each node spends transmitting while making full use of the multiuser detector capability.

We therefore propose to extend the RP-CDMA Phy with two additional features: Simultaneous Transmission and Payload Channel Acknowledgements. We discuss these features in the following sections, after which we will be in a position from which we can describe our MAC protocol and acknowledgement mechanisms.

4.2 Simultaneous Transmission

Simultaneous transmission is motivated by the observation that there is no requirement for packets arriving concurrently at a receiver to have been transmitted by different senders. Furthermore, when we consider that a multiuser detector can receive multiple packets simultaneously it makes sense to pair this capability with an equally capable transmitter. We therefore propose to exploit the capability of the multiuser detector by having a transmitting node compute and transmit the aggregate waveform of multiple packets, each with unique payload spreading codes and power levels. In this way, a transmitter can use a single frequency band to separate data in both time and transmission power. The only restriction is that packet headers be staggered so as to not overlap and collide, and that a node not transmit a number of simultaneous packets beyond the capability of the multiuser detector, K, since there is no advantage in transmitting more packets simultaneously than the receiver is able to successfully decode. Aside from these restrictions, we are free to transmit multiple packets concurrently to one or many network peers.

Simultaneous transmission is effectively the inverse of multiple packet reception, but instead of the receiver decoding multiple packets simultaneously using their unique spreading codes and iterative cancellation, the transmitter encodes multiple packets simultaneously using unique spreading codes and signal aggregation. By assigning each packet a power level and then computing the aggregate waveform of all packets, the transmitter is able to concurrently spread multiple packets over the range of transmitter power capability. On the receive side, the multiuser detector uses iterative cancellation to successively decode each packet from most powerful to least powerful. In this way, a single transmitter can send up to K packets simultaneously - where K is the capability of the multiuser detector - as long as the packet headers are staggered so as to not overlap. Because each packet is encoded using a unique spreading code, there is no restriction on the intended recipients of these aggregated packets, as the multiuser detectors at each other network node will be capable of decoding all of the simultaneous packets in the same manner as if the packets had arrived from different transmitters simultaneously.

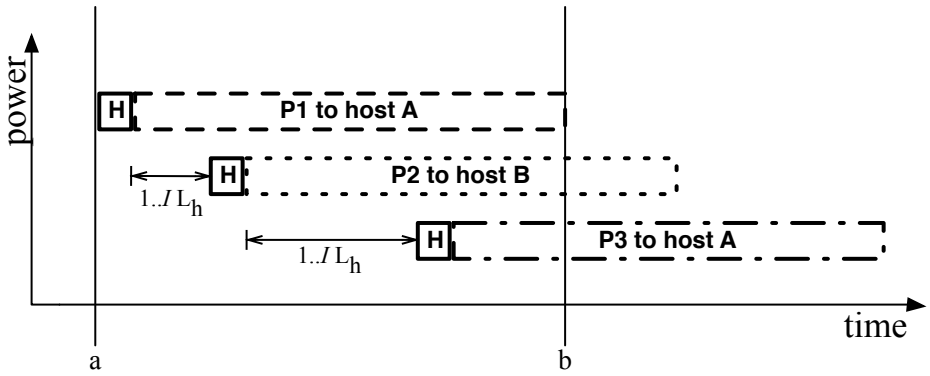


Figure 4.1: Simultaneous transmission of multiple packets to one or more receiving nodes, with spreading code indicated by line type. Note that only packet headers share a spreading code, and are therefore staggered so as to not overlap by some number of packet header intervals, denoted by L_h. In a given frequency band, we distribute data in both time and power, which allows us to transmit multiple packets in the same time it takes to transmit just one packet, denoted here between points a and b.

Simultaneous transmission is illustrated in Figure 4.1, which shows three packets being transmitted simultaneously. Each packet payload is encoded with a unique spreading code, indicated by the line type, and is labelled with the intended receiver of the packet. Figure 4.1 also illustrates how simultaneous transmission enables us to reduce the time a node spends transmitting while increasing the amount of data transmitted, because it enables a single transmitter to fit more data into unit time. Note that the transmitter is able to commence transmission of packets P2 and P3 before the transmission of P1 is complete, which means that both P2 and P3 will finish transmission earlier than they would if they were transmitted one at a time. Being able to fit more data into unit time allows each node to reduce the amount of time spent in the transmit state, which we expect will reduce the number of packets lost due to the half duplex nature of the radio. At the same time, by fitting more data into unit time, more time is available to other nodes in the system for their turns transmitting.

The risk with simultaneous transmission is that in the event of two or more nodes beginning transmission at the same time many packets may be lost to header collisions rather than just one or two. This risk can be illustrated if we imagine the classic hidden node problem, where two sending nodes share a receiving node

but cannot communicate with each other. Without simultaneous transmission, if the two sending nodes each transmit one packet such that their headers collide at the receiving node then both packets will be lost. With simultaneous transmission, the same situation could result in up to K packets lost from each sending node, or $2K$ packets in total.

In order to mitigate this risk, we stagger simultaneous packet headers by a random number of packet header intervals, which we denote by L_h.[2] After transmitting a packet header the transmitter waits for a number of header intervals in the range $[1, I)$, and then begins transmitting the next packet. Now if two sending nodes begin transmitting at exactly the same time to a common receiving node, their first packets will collide at the receiving node, but because we randomly stagger each subsequent simultaneous packet it is possible that their headers will interleave and be successfully received. Thus, in the case where two nodes begin transmission at the same time, traffic from either of them destined for third parties may still be recoverable. This is why we randomly stagger our packet headers rather than transmit all of the packet headers in a block.

Simultaneous transmission is our primary solution to the transmitter bottleneck we observed in our early RP-CDMA experiments. By sending several packets concurrently, a single sending node is able to exploit the full capability of the multiuser detector at the receiver while reducing the time required to send a given amount of data.

4.3 Payload Channel Acknowledgements

Payload channel acknowledgement refers to the practice of encoding both the packet header and payload of an acknowledgement packet (Ack) using the payload spreading code of the packet being acknowledged. This is in contrast to the standard RP-CDMA procedure of encoding the packet header using a common code, and the payload with a random code. Because the transmitter knows which payload codes it used to send packets to its neighbours, it can listen for acknowledgements us-

[2] A packet header interval is just the time it takes to transmit one packet header, which we saw in Figure 3.3 has $L_h = 144$ b $= 18$ B.

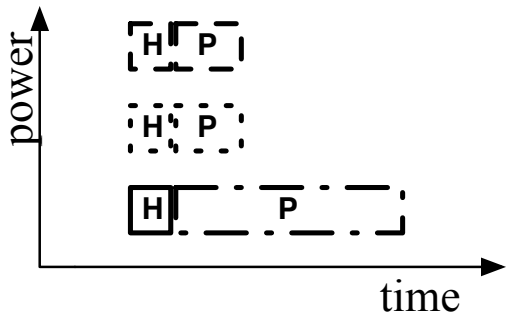


Figure 4.2: Payload Channel Acknowledgements. By sending both the header and payload of an Ack packet using the payload spreading code of the packet being acknowledged, load is reduced on the common header channel. In this figure, the type of line used to depict the packet headers and payloads represents their different spreading codes. Because the packet headers for Ack packets are sent using a payload code instead of the common header code, multiple Ack packets can be sent concurrently.

ing those same codes, and both the header and payload of the acknowledgement packet can be encoded using this payload code. In this way, acknowledgments are sent entirely in the packet payload channel, and do not impact the common header channel. Furthermore, when combined with simultaneous transmission, up to K acknowledgement packets can be transmitted simultaneously with no requirement to stagger the packet headers. As a result, a transmitter can send up to K acknowledgements in the same time it takes to send one, which significantly reduces the load that acknowledgement packets place on the transmitter and network.

This is depicted in Figure 4.2, where we show two Ack packets and one data packet being sent simultaneously. Again, the spreading code used to transmit a particular packet header or payload is indicated by line type. Because Ack packet headers are each sent with the same code as their payload, there is no requirement to stagger Ack packet headers.

Payload channel acknowledgements further reduce the time spent by a node in the transmit state, which allows each node more time to transmit new data. This further enables our MAC to minimize the amount of time spent transmitting, and by reducing the number of headers sent in the common header channel, decreases the probability of packet header collisions.

4.4 MAC Protocol

We can now begin our discussion of our MAC protocol. Because we are interested in simple MAC protocols that do not require coordination between nodes, we consider a simple random backoff based MAC. This MAC is similar in spirit to the 802.11 backoff mechanism, but whereas 802.11 uses a binary exponential backoff algorithm to determine how long to wait after failing to gain the channel, we use a simple uniformly random backoff interval with no feedback or adjustment in the event of successful or failed transmission. Our backoff mechanism is therefore simpler than that in 802.11 [48].

Our MAC maintains a drop tail queue of packets which have arrived from the network layer but have not yet been transmitted. This queue can hold up to M packets, and any packets which arrive while the queue is full are dropped. Once a packet enters the queue, the MAC begins the process of trying to send it. When having a packet to send, the MAC first checks the Phy to determine if it is 'safe' to transmit and, if so, sends the packet to the Phy for transmission after waiting a random number of packet header intervals in the range $[1, B)$. If the Phy starts receiving a packet while the MAC is waiting, then the transmission is postponed until the transceiver becomes available again, at which time the MAC repeats the random backoff procedure.

Algorithm 4.1 shows the Phy level evaluation of whether it is 'safe' to transmit. This algorithm simply returns true if the Phy is idle or if it is currently transmitting fewer than K packet payloads, and otherwise returns false. It is worth noting here that we are using the Phy state to find our solution to the problem of determining whether or not the channel is occupied at any given time. A more common solution to this problem is to perform a clear channel assessment (CCA) by estimating the noise floor in the channel and comparing that estimate with the current channel conditions [48]. In our solution, we consider the channel to be unoccupied if the Phy is not actively receiving any packets. This means that we are not performing a CCA in the same manner as 802.11, but are rather substituting the CCA function for the activity status of the Phy. This is conceptually simpler than trying to guess what the

noise floor is in the clear 802.11 channel, and also practically simpler since the Phy knows directly whether or not it is currently processing any incoming packets. The potential risk of assessing whether or not it is safe to transmit at a given time in this manner is that it may ignore evidence that there is traffic in the channel that has not been picked up by the receiver. This eventually depends on the properties of the radio receiver, such as the average number of header synchronization bits required before the receiver acquires the timing of a particular signal or the packet encoding scheme chosen, but we proceed on the assumption that if the radio is actually capable of reliably detecting and synchronizing on an incoming transmission, then our mechanism of simply querying the Phy for its status would be sufficient in a real implementation, particularly since the RP-CDMA system is highly tolerant of overlapping transmissions.

Algorithm 4.1 Phy CanTx()

MPR is the multiuser detector, with capability K
$TxList$ is the list of packets currently being transmitted
$State$ is the state of the transmitter.

if $State =$ IDLE **then**
  return true
end if
if $State =$ TX-PAYLOAD **and** $TxList$.size $\leq$ *MPR*.K **then**
  return true
end if
return false

Algorithm 4.2 shows the algorithm for selecting a random backoff period before beginning a transmission. In order to support simultaneous transmission, this function checks the Phy for its state and returns a random time in the range $[1, B)$ header intervals initially, or in the range $[1, I)$ for simultaneous transmission.

Algorithm 4.3 puts these two together into the algorithm that executes whenever the device has data to send, and is visually illustrated in Figure 4.3. In this diagram, when the MAC has a packet waiting to be sent from its queue, it enters the diagram at the top. It first checks if the Phy is ready to transmit. If not, then the MAC will wait until the Phy notifies it that it is available to transmit. Once the Phy is available, then the MAC waits a random number of packet header intervals in

Algorithm 4.2 MAC CalculateBackoffTime()

B is the initial maximum number of header intervals to wait
I is the maximum simultaneous transmission header intervals
L_h is the length of a packet header
S is the channel speed
Phy is the device Phy

if Phy.State = TX-PAYLOAD **then**
  return Rand(1,I) $\cdot L_h/S$
end if
return Rand(1,B) $\cdot L_h/S$

the range $[1, B)$, then checks again if the Phy is still available. If the Phy is still available to transmit after the backoff period, then the MAC forwards the packet to the Phy for transmission. The Phy will then begin transmitting the packet into the channel.

Algorithm 4.3 MAC SendFromQueue()

$Data$ is the queue of packets that need to be sent
Phy is the device Phy layer

while not $Data$.Empty() **do**
  while not Phy.CanTx() **do**
    Wait()
  end while
  Wait(CalculateBackoffTime())
  if Phy.CanTx() **then**
    $Next \leftarrow Data$.PopFront()
    Phy.Send($Next$)
  end if
end while

We implement the random backoff period in order to resolve header channel contention in the event of several neighbouring nodes all wishing to transmit at the same time. This is easily illustrated with a simple example with three neighbouring nodes, depicted in Figure 4.4. Suppose that nodes A, B and C are neighbours and can all communicate with each other. Suppose that node C has a packet to transmit to node A, and both nodes A and B have packets for each other. If node C begins transmitting its packet (4.4a), both nodes A and B will wait until C's transmission

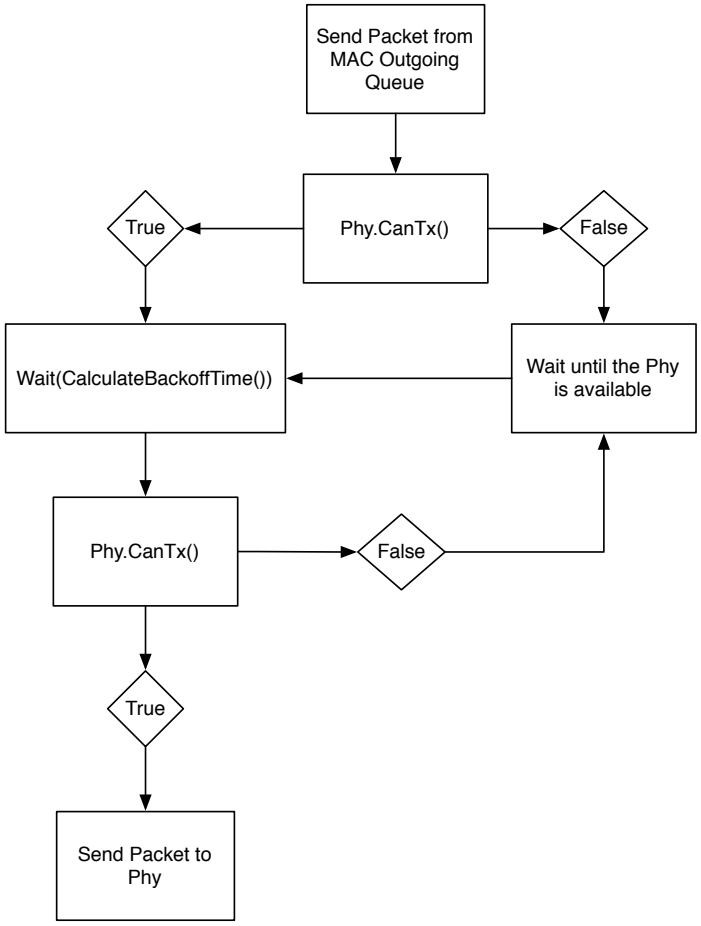
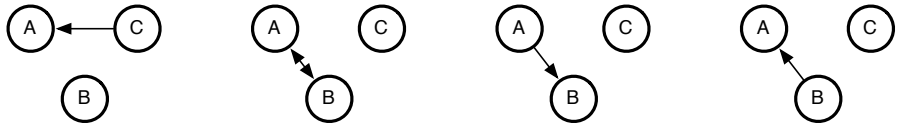


Figure 4.3: MAC Sending Algorithm. When the MAC has a packet to send, it first checks of the Phy can transmit. If so, it waits a random number of packet header intervals, then sends the packet to the Phy for transmission.



(a) C sends to A

(b) Both A and B send, causing loss

(c) A sends after random backoff

(d) B sends after random backoff

Figure 4.4: An example showing why we implement a random backoff algorithm in our MAC. After C finishes sending to A (a), if both A and B were to commence transmission immediately then their packets would both be lost (b). If both A and B choose a random backoff interval, then A can send into the channel first (c), and then B afterwards (d).

is complete before sending their packets. As soon as the transmission from node C ends, the MACs in nodes A and B will be notified that their Phys are no longer receiving any packets. At this point, if our MAC protocol did not wait before sending their packets to the Phy, then both nodes A and B would immediately begin transmitting their packets to each other. Since we have half duplex radios, neither A nor B would receive the packets from one another, and both packets would be lost (4.4b). However, if both nodes A and B randomly select a backoff period, then there is some probability that they select different random periods and one of them will begin transmitting first. Suppose node A begins transmitting first (4.4c). In this case, nodes B and C would wait until the transmission from A was complete, after which node B would randomly wait for some time period, and then would commence transmission to node A (4.4d). Thus, the random backoff period prevents nodes from all trying to transmit at once as soon as a transmission finishes.[3] Our random backoff period then is designed to facilitate orderly access to the channel by one node at a time. Having only one transmitting node at a time is important in ad hoc networks because a node may have packets for one or several of its neighbours, and so it is best if only one node transmits at a time in a given set of neighbours. We use a random backoff period because it does not require the nodes to coordinate their channel access, which was one of our design goals from Section 1.2.

[3]The problem of many entities all trying to do the same thing at once, resulting in nothing being accomplished, is sometimes called the Thundering Herd Problem. `http://en.wikipedia.org/wiki/Thundering_herd_problem`

We have seen here that our MAC is conceptually straightforward. We apply a uniform random backoff mechanism before initiating a transmission, and once transmitting utilize simultaneous transmission to send as much data as possible in given time. Having described how our MAC works, we turn our attention to acknowledgement mechanisms.

4.5 Acknowledgements

In our work, effectively addressing the acknowledgement problem came down to answering the question of *when* to ack. In a system designed with the limiting constraint that a single node can only receive one packet at a time from only one transmitter, such as 802.11, it seems obvious that the acknowledgement should come immediately after the single packet was successfully received. In the RP-CDMA based system we propose here, a node may be receiving several packets concurrently - perhaps from different sending nodes - and it becomes problematic to send an acknowledgement after each packet is received because switching to the radio transmit state necessarily means dropping any packets that are not fully received.

Our focus when developing acknowledgement policies in this work was therefore much more on solving the problem of when to send acknowledgements, rather than how. Our acknowledgement policies thus send an acknowledgement for each successfully received packet. This may seem problematic at first glance, but when we consider that payload channel acknowledgements and simultaneous transmission remove ack traffic from the common header channel and compress sending many ack packets into the time required to send only one, we see that acknowledging every packet is an algorithmically simple solution that actually has relatively little impact on the system.

The problem of when to ack is illustrated in Figure 4.5. In this figure we see that if the receiving node sends an acknowledgement after the first packet, $P1$, is received at time a then the second and third packets, $P2$ and $P3$, will be lost because the radio is half-duplex. Similarly, if we choose to send an acknowledgement after $P2$ then $P3$ will be lost. We can extend this reasoning to the general case and see

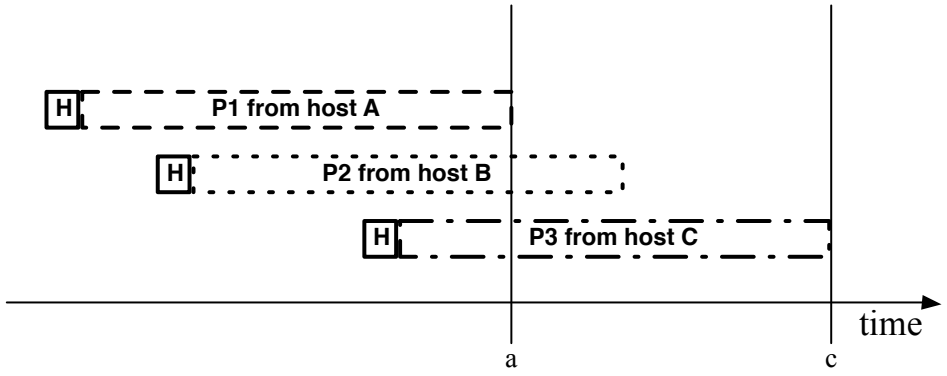


Figure 4.5: When to send acknowledgements? If an acknowledgement for $P1$ is sent at time a, then $P2$ and $P3$ will be lost because the radio is half-duplex. If the receiver waits until time c to send an acknowledgement, then host A will be left waiting in the interval $[a, c]$.

that for any fixed timeframe or number of packets that a receiving node waits before sending acknowledgements can yield a case where some packets are dropped as the radio switches to transmit. Thus, trying to guarantee that an acknowledgement will be sent in some fixed period will inevitably lead to lost packets at the receiving node. In the multiuser detector environment, the receiving node therefore benefits from being able to wait for longer periods before sending acknowledgement packets.

At the sending node, we have an additional problem. If the sending node knows that the receiving node will guarantee acknowledgement within some fixed timeframe after receiving a packet, then it must refrain from transmitting within this timeframe or else it may miss the acknowledgement. This time spent waiting for acknowledgements causes delay in the transmission of subsequent packets. In this situation, the sending node can minimize delay of subsequent packets if the receiving node sends an acknowledgement as soon as possible. As system load is increased the receiving node's benefit from longer delays before transmitting acknowledgements conflicts with the sending node's benefit from minimal delays before receiving them.

We see then that trying to choose a fixed timeframe in which to send an acknowledgement becomes an effort in balancing the needs of the receiving and sending nodes. The receiving node prefers to wait and receive as many concurrently arriving packets as possible before transmitting acks in order to minimize dropped packets,

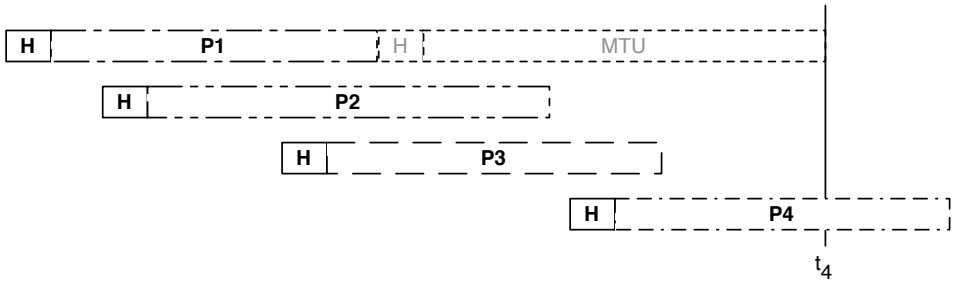


Figure 4.6: Immediate Ack will wait up to 1 MTU time period after P1 arrives before sending an acknowledgement. Adopting this policy allows packets P2 and P3 to be received, but P4 will be lost.

and the sending node prefers that the receiving node transmit an acknowledgement as soon as possible in order to minimize waiting time and wasted bandwidth.

With this in mind, we can identify two main classes of acknowledgement policy: those with fixed time acknowledgements, and those with non-fixed time acknowledgements. In a fixed time system, the receiving node will guarantee acknowledgement within some fixed timeframe of having received a packet, and in an non-fixed time system the receiving node makes no such guarantee. In our work we refer to these policies as Immediate Ack and Eventual Ack, respectively. We describe Immediate Ack in Section 4.5.1, and Eventual Ack in Section 4.5.2.

4.5.1 Immediate Ack

Immediate Ack implements the intuitive practice of sending a packet and then waiting for an acknowledgement. If an acknowledgement is not received in some fixed timeframe, which we call the *Acktime*, then the packet is retransmitted. Packets are retransmitted some number of times, R, before being dropped.[4] The receiving node, therefore, must guarantee that an acknowledgement is sent before the *Acktime* expires in the sending node.

When combined with simultaneous transmission, the sending node sends one or more packets concurrently, but as soon as transmission of the first packet is complete the device stops adding new packets, completes all simultaneous transmissions, and begins waiting for acknowledgements from the receiver(s). For each

[4]We set a default number of retries, $R = 7$, in our experiments, which follows the 802.11 standard.

sent packet, if an Ack is not received in the *Acktime* then the sending node retransmits the packet immediately, up to R times. After receiving a packet the receiving node tries to send an Ack immediately, but will delay for a time period long enough to complete reception of a single maximum sized packet, which is just the device MTU divided by the channel speed. This is illustrated in Figure 4.6, where the imaginary packet labelled MTU indicates the amount of time the receiving node will allow before switching to transmit and sending acknowledgements for P1, P2, and P3. This waiting time permits the receiving node to finish reception of any packets from the same sending node that were sent simultaneously with the first packet before sending acknowledgements. If packets are still being received after waiting for the MTU period (from a third node), the receiving node will break off packet reception in order to send Acks. From the receiving node point of view this acknowledgement policy implements a guaranteed fixed time after receiving a packet in which an acknowledgement will be sent, while still allowing several packets to be received concurrently. From the sending node point of view, the Immediate Ack policy may be thought of as a type of 'Stop and Wait', since the sender stops sending and waits for acknowledgements after sending out some number of simultaneous packets along with the first packet.

The choice for *Acktime* therefore should be longer than the minimum time in which a receiving node could possibly complete the first packet reception, wait one MTU period, and then transmit an Ack. In practice we also add a small random time period to the standard *Acktime* when deciding how long to wait before retransmitting an unacknowledged packet. This small random time is meant to avoid replaying packet header collisions. What we mean here is if it happens that two nodes begin transmitting such that their packet headers collide at a receiver, then those packets will be lost. If both nodes then retransmit their packets after waiting for the *Acktime*, then the packets will collide again at the same receiver. In this situation, the two sending nodes will deterministically collide with their retransmissions each time they retransmit. In order to avoid this situation, each sending node adds a small random time value to the *Acktime*, which serves to slightly randomize node retransmission time. When retransmitting several packets, a single node must

be careful to not schedule retransmissions such that the packet headers overlap, and so each node ensures that its scheduled retransmission times are strictly increasing.

We are now in a position where we can write our algorithms for sending node retransmission and receiving node acknowledgement transmission. We show these algorithms in Algorithms 4.4 through 4.7. We begin with the algorithm used in the sending node when a packet is sent from the MAC down to the Phy for transmission, the Immediate Ack OnSend(), Algorithm 4.4. This algorithm sets the packet retransmission counter to zero and calls the ScheduleRetransmission() Algorithm 4.5 to schedule the retransmission time for the packet, p. When the scheduled event fires, the packet is immediately retransmitted via the Retransmit() Algorithm 4.6. If an acknowledgement for a packet arrives before the scheduled retransmission, then the retransmission event is cancelled.

Algorithm 4.4 Immediate Ack OnSend(p)

p is the packet just transmitted

p.retransmissions $\leftarrow 0$
ScheduleRetransmission(p)

Algorithm 4.5 Immediate Ack ScheduleRetransmission(p)

p is the packet to schedule retransmission for
min_retry_time is the earliest a retransmission can be scheduled
$Acktime$ is the minimum waiting time before retransmitting
B is the maximum initial backoff period
L_h is the packet header length
S is the channel speed

$X \leftarrow$ Rand(1,B) $\cdot L_h/S$
$W \leftarrow$ Now() $+Acktime + X$
if $W < min\_retry\_time$ **then**
  $W \leftarrow min\_retry\_time + X$
end if
$min\_retry\_time \leftarrow W + L_h/S$
Schedule(W, Retransmit(p))

Upon receiving a packet, a receiving node will attempt to send an Ack within the fixed time it would take to receive a MTU sized packet. This is shown in Algorithm 4.7, which details the algorithm used when a data packet is received.

Algorithm 4.6 Immediate Ack Retransmit(p)

p is the packet to retransmit
R is the maximum number of retransmission attempts

p.retransmissions $\leftarrow$ p.retransmissions + 1
Phy.Send(p)
if p.retransmissions $< R$ **then**
  ScheduleRetransmission(p)
end if

Algorithm 4.7 Immediate Ack DataReceived()

p is the data packet received
Dev is the network device
Phy is the network device Phy
S is the channel speed

$MaxWait \leftarrow$ Now() + Dev.MTU / S
while not Phy.CanTx() **and** Now() $< MaxWait$ **do**
  Wait()
end while
Phy.Send(p.Ack)

These algorithms constitute our fixed time acknowledgement policy. We now describe our non-fixed time acknowledgement policy.

4.5.2 Eventual Ack

Whereas the Immediate Ack policy is designed to guarantee to the sender that an acknowledgement will be sent for a received packet within some fixed timeframe, the Eventual Ack policy is designed to minimize the disruption that acknowledgements have on the receiving network device. In terms of when to send acknowledgements, as illustrated in Figure 4.5, the receiving node chooses to send acknowledgements whenever all packet receptions are complete, and so will never break into a packet reception to send an acknowledgement. At the sending node, it does not make any sense to wait indefinitely for an acknowledgement, and so the sending node continues sending data normally, trusting that past packets will be acknowledged whenever their recipients have an opportunity to get around to it.

In implementation, this acknowledgement policy is simpler than the Immediate

Ack policy. At the receiving node, waiting until all packet receptions are complete is equivalent to waiting for the channel to become free, and so Ack packets can be inserted into the regular outbound data queue, but given priority over ordinary data. At the sending node, data packets are simply pushed out as soon as possible. The primary difficulty for the sending node is deciding when a packet has been lost. Because a packet acknowledgement may arrive at effectively any time, the absence of an acknowledgement packet does not necessarily mean that the packet was lost, only that its acknowledgement was not yet sent.

To enable a node to accurately identify lost packets, we can rely on the ordering of sent and acknowledged packets. Each node maintains four ordered lists: Data, Retries, Acks, and Sent. In the sending node, when a packet is passed to the device MAC from the network layer for transmission, it is put into the Data list, and after being sent to the Phy layer and transmitted the packet goes into the Sent list. At the receiving node, when a data packet is received an Ack is put into the Acks list. Whenever the device assesses that it is safe to transmit, as determined by the MAC SendFromQueue() Algorithm 4.3, then the device will transmit packets from the Acks, Retries and Data lists in that order. In this way, Ack packets are prioritized first, Retry packets second, and new packets last.

Because Ack packets are put into the Acks list in the same order in which their data packets were received and the sender maintains the Sent list in the same order in which packets were sent, the sequence of Ack packets from a receiving node should therefore be in the same order as the corresponding packets in the sending node Sent list. This enables the accurate identification of lost packets by the sending node: After getting an Ack from a receiving node, the sending node can mark as lost any packet that was sent to the same receiving node and is between the front of the Sent list and the acknowledged packet. Lost packets are put into the Retries list, and upon retransmission are moved to the end of the Sent list to preserve the ordering.

The Eventual Ack policy slightly modifies the SendFromQueue() function in the MAC to support sending packets from each of the Acks, Retries and Data queues in priority. Algorithm 4.8 shows the modification, where we simply send

data from each of the queues in order. In this version, SendFromQueue() will transmit Ack packets until all waiting Acks are sent or the multiuser detector limit of K is reached. Afterwards, if it is still clear to transmit after sending Acks, then the device will transmit one Retry or one new Data packet. Note that the device continually checks the Phy CanTx() method to determine that it is still safe to transmit, since each packet sent may change the state of the device. In particular, after sending a Retry packet, the next call to the Phy CanTx() method will return false because the device will be transmitting the retried packet header.

Algorithm 4.8 Eventual Ack SendFromQueue()

```
Data, Retries, Acks, Sent are ordered lists.
Phy is the device Phy layer

while not Data.Empty() and not Retries.Empty() and not Acks.Empty() do
  while not Phy.CanTx() do
    Wait()
  end while
  Wait(CalculateBackoffTime())
  while Phy.CanTx() and not Acks.Empty() do
    Phy.Send(Acks.PopFront())
  end while
  if Phy.CanTx() and not Retries.Empty() then
    Next ← Retries.PopFront()
    Phy.Send(Next)
    Sent.MoveToBack(Next)
  end if
  if Phy.CanTx() and not Data.Empty() then
    Next ← Data.PopFront()
    Phy.Send(Next)
    Sent.PushBack(Next)
  end if
end while
```

Algorithm 4.9 shows how the device identifies previously sent packets as having been lost. This procedure simply iterates through the Sent list looking for a match to the Acknowledged packet. Any packets that were sent to the same receiving node before the Acknowledged packet are marked for retransmission, since the ordering of the Sent and Acks lists ensures that an Ack would have already been sent had the packet been received successfully.

Algorithm 4.9 Eventual Ack ReceiveAck(Ack)

Ack is the Ack just received

```
for i = Sent.First to i = Sent.Last do
  if Sent[i].Destination = Ack.Sender then
    if Sent[i].Sequence = Ack.Sequence then
      Sent.Erase(i)
      return
    else
      Retries.PushBack(Sent[i])
    end if
  end if
end for
```

The only remaining case to handle is the instance where all of the packets sent to a particular receiving node are lost. If this happens, then the sending node will never receive any Acks for any of those packets, and will therefore never identify those packets as lost using Algorithm 4.9. To guard against this, we can employ a very long timer in the sending node which, when it expires, marks a packet for retransmission. Thus, in Eventual Ack, we employ very long values for *Acktime*, and when a packet remains in the Sent list for longer than *Acktime* it is put into the Retries list.

4.6 Summary

In this chapter we have described our RP-CDMA network device MAC layer. We began by motivating our design decisions, and specifically noted that the addition of the multiuser detector in the receiver changes the system performance limitation from that of packet reception capability to packet transmission capability. This observation motivated our two extensions to the RP-CDMA protocol, simultaneous transmission and payload channel acknowledgements, which aim to pair the powerful multiuser detector with an equally powerful transmitter in the ad hoc network environment. We then described the algorithms which determine how our MAC decides when to transmit, and finally discussed the acknowledgement problem in the presence of multiuser detectors. Particularly, we found that the possibility of receiv-

ing several overlapping packets simultaneously complicates the decision of when to send acknowledgement packets. Finally, we outlined two acknowledgement policies which we implemented in our system, one which guarantees acknowledgements in a fixed timeframe after packet reception, and one which does not.

Chapter 5

Experimental Setup

The goal of our performance study is to evaluate the reliability and performance of our RP-CDMA network device in the multihop ad hoc context, so our experiments were designed to allow us to perform this evaluation. The ns-3 simulator and our RP-CDMA MAC protocol have several configurable parameters which we describe here, and at the end of each experiment we measured the performance of the system. In this chapter we will describe each of our configurable parameters, whether they were fixed or varied between experiments, and what output we measured to evaluate performance.

Each of our experiments consisted of some number of nodes placed in a two dimensional field with sufficient spacing between them such that the network is connected but not fully connected. This ensured that there was some degree of multihop required for packets to traverse from one end of the network to the other. During experiments each node generated data traffic with exponentially distributed packet inter-arrival times and uniformly distributed destination node.

In each experiment we measured the number of packets which entered the network and packets which arrived at their final destination. For each packet arriving at its final destination, we recorded the time it took to traverse the network from its originating node. At the end of each experiment, we used the aggregate statistics of all the packets which successfully arrived at their final destination to determine how many packets were lost, average system throughput and average end to end delay.

For ease of reference, we show all of the fixed values, variables, and measured values in Table 5.1. In this table, the fixed parameters are set globally for all exper-

iments, and the variable parameters are the system parameters we will investigate in our performance study. The measured values are our performance metrics.

The remainder of this chapter describes the setup and configuration of our experiments in detail. Our aim is to enable the reader to create similar or identical experiments in ns-3, and so we will discuss some of the implementation details in the simulator. We begin with a discussion of the simulator configuration in Section 5.1, followed by the node configuration and network topology in Section 5.2. We then discuss how the nodes behaved during each experiment, and particularly how they generated traffic in Section 5.3. We discuss system calibration and experiment duration in Section 5.4 and then how we measure system performance in Section 5.5. Section 5.6 describes a 2^k factor analysis investigating the relative impacts that our variable system parameters have on performance.

5.1 Simulator Configuration

We describe here the configuration of the ns-3 simulator, with the aim of providing sufficient detail to enable the independent replication of our results.

Random numbers were generated using the built in ns-3 random number generator (RNG), which itself is based on the MRG32k3a generator by LEcuyer *et al.*[31]. This is the same RNG as that used in ns-2, and features a large number of independent streams, each of which has a large number of substreams. Specific streams and substreams can be selected via seeds and run numbers. Each random number variable in ns-3 uses its own RNG. Since substreams of the same stream do not overlap, we used a single seed and advanced the run number to produce independent trials. In all of our experiments, we used a seed value of 12345 and incremented the run number from 1 to the number of trials we wished to perform. We performed most of our experiments with 10 trials, and therefore usually used run numbers 1 through 10. Experiments with significantly more trials did not appreciably affect our results, and so we considered 10 to be sufficient for our purposes. Our experiments are therefore reproducible by setting the seed and run number to the same as the trial we wish to reproduce.

Fixed Parameters	
Parameter	**Description**
RNG Seed	The random number generator seed. Fixed for all experiments.
RNG Substream	The random number generator substream id. Changed for independent replications.
Channel Speed	The data rate of the wireless channel.
Number of Nodes	The number of nodes in each experiment.
Runtime	The duration of each experiment.
UDP Data Size	The data packet size in UDP data experiments.
TCP Data Size	The data stream size transferred in the TCP experiments
Variable Parameters	
Parameter	**Description**
Network Topology	The network topology. Either Grid or Random.
Offered Load Per Node	The load offered to the network by each network node. Determined by data arrival rate, λ, and UDP/TCP Data Size.
Ack Policy	The acknowledgement policy used, as described in Section 4.5.
Acktime	The elapsed time before a sending node MAC will retransmit an unacknowledged packet.
I	Upper bound for the simultaneous transmission inter-packet backoff time, as described in Section 4.2.
B	Upper bound for the initial backoff time, as described in Section 4.4.
M	Maximum size of the MAC data queue, in packets.
K	Capability of the multiuser detector. The maximum number of packets that the detector can resolve concurrently.
Measured Values	
Measured Value	**Description**
Percent Packets Lost	The fraction of packets that were sent down by the Sender application, but not received at the destination Receiver application.
End to End Delay	For those packets that arrived at their final destination Receiver application, how long they took to traverse from Sender to Receiver.
System Throughput	The total amount of data that successfully traversed the network from Sender to Receiver, divided by the experiment runtime.

Table 5.1: System Parameters and Measurements

Network routing was via the Optimized Link State Routing (OLSR) [11] and Static routing implementations from ns-3. Since our nodes were static, we did not require a more sophisticated routing algorithm, such as Ad-hoc On-Demand Distance Vector (AODV). We are not specifically concerned with evaluating the performance of the network routing algorithm in this work, and therefore chose OLSR and Static because they are effective with relatively low overhead. With the combination of Static and OLSR routing, node routing tables would be populated at the beginning of each experiment via OLSR broadcasts, and thereafter nodes would find routes via their Static entries. OLSR did continue to broadcast routing information throughout the experiments, even though the routing tables did not change after their initial population. We consider this routing traffic in the channel to contribute to the practical realism of our simulations.

Channel speed was set to 1 Mbps, which was chosen because it facilitates easy comparison to 802.11, which uses a simple Differential Binary Phase Shift Keying (DBPSK) Direct Sequence Spread Spectrum (DSSS) implementation at this speed [48]. We assume a relatively simple BPSK DSSS CDMA modulator in our RP-CDMA system, which can be compared to the 802.11 equivalent in a straightforward manner when working with channel capacity and bandwidth. We acknowledge the existence of more sophisticated modulation and symbol encoding schemes, such as those used for high speed 802.11n [49] or those employed in various multiuser detectors [44, 25], but these can be layered on top of the underlying CSMA and RP-CDMA based devices without changing the fundamental comparison. We therefore choose the simplest implementation of each device and set the channel speed to 1 Mbps.

5.2 Node Configuration and Topology

Our experiments focused on two node configurations: grid and random. In the grid topology, our aim was to provide a regular network topology in which individual nodes had an approximately equal number of neighbours. To this end, we designed our grid such that nodes could communicate with their neighbours along the same x

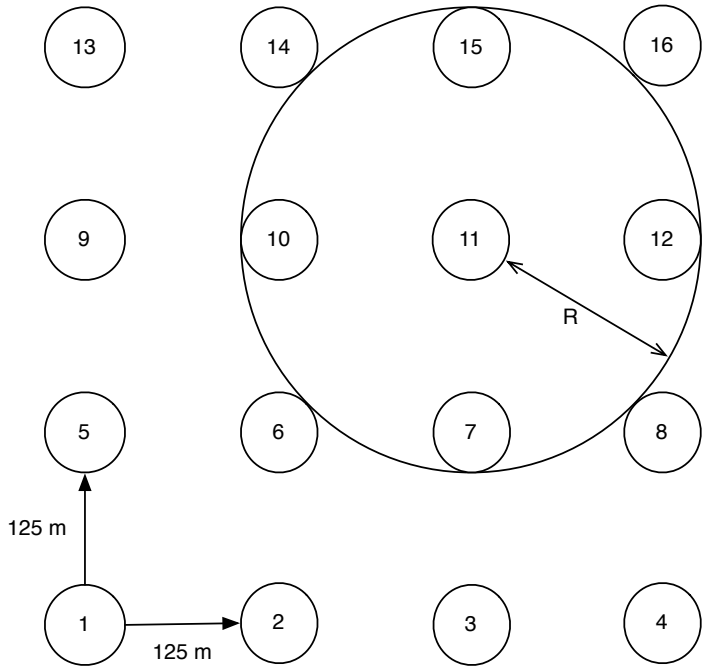


Figure 5.1: Grid Topology Configuration. In this topology, nodes can communicate with their neighbours on the same x and y axis, but cannot communicate diagonally. This is illustrated with the node communication radius R, centred on node 11.

and y axis, but could not communicate diagonally. This is illustrated in Figure 5.1. In this figure, the node communication radius is illustrated with the circle of radius R centred on node 11. We used a 4×4 grid of nodes in all of our grid experiments, as this offered a network with a maximum of 6 hops on which experiments would complete on available hardware in a reasonable period of time.

The distance between nodes of 125 m was chosen empirically. We performed experiments over a range of node spacings and measured the packet loss as we increased the distance between nodes. From these results we selected a distance that would allow nodes to communicate along the x and y axes, but not diagonally. We show our packet loss results in Figure 5.2. In this figure, we can see that as the inter-node spacing increases to 160 m that system packet loss increases to 100%, indicating that we have surpassed the effective range of the nodes. We can see that if we space the nodes at 125 m, then the diagonal distance between nodes is $\sqrt{125^2 + 125^2} \approx 177$ m, which is beyond the maximum transmission distance.

We derive the same result analytically using the device default transmit power,

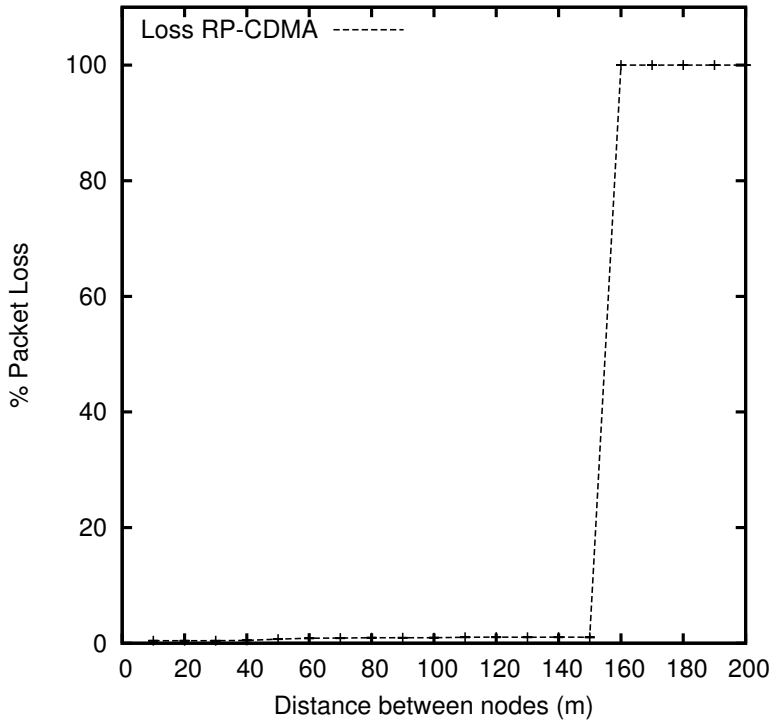


Figure 5.2: Grid topology effective range. As we increase the distance between nodes, we see that packet loss increases to 100% when the inter-node distance is too large. From this we can estimate the effective range of the nodes.

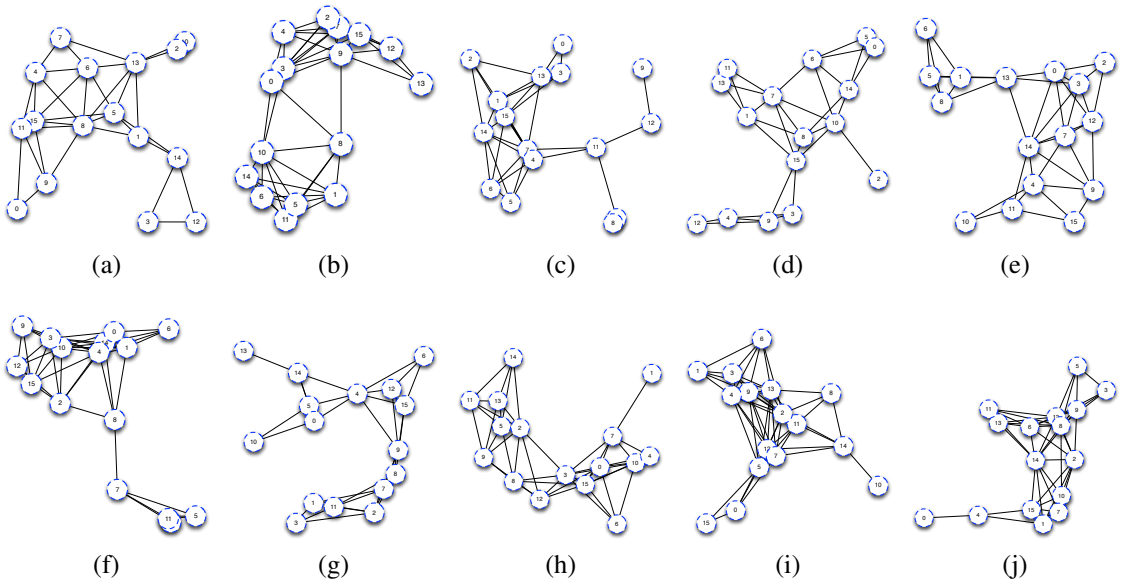


Figure 5.3: The 16-node random topologies tested.

our loss propagation model, and the device detection threshold. We use a default transmit power of 16.0206 dBm and a detection threshold of -96 dBm, and we can solve for the maximum distance using Equation 3.2:

$$\begin{aligned} d &= d_{ref} \cdot 10^{\frac{P_{tx}-P_{rx}-L_{ref}}{30}} \\ &= 1.0 \cdot 10^{\frac{16.0206+96-46.6777}{30}} \\ &= 150.6942 \end{aligned}$$

Which agrees with our observation that packet loss increases to 100% after 150 m. Note that we have chosen our default transmit power and detection threshold to match those of the ns-3 802.11 model, which facilitates comparison between the two.

In addition to our grid topology, we also experimented with random topologies. In these experiments we generated 10 different random connected topologies with the same average node density as our grid topology, so 16 nodes inside a 375 $\times$ 375 m box. Node placement was uniformly random in both x and y. These topologies are shown in Figure 5.3.

Nodes were static throughout each experiment, since we are interested here in

evaluating the effects of the reliable link on packet loss and system performance, which does not require node mobility.

5.3 Node Behaviour

Each simulation started with a 200 second warm up period in which nodes identified routes through the network to each other node. In these 200 seconds, the OLSR routing protocol exchanged routing information with each neighbour several times, until the routing table was fully populated. OLSR continued to update the routing tables throughout the experiment, but because the nodes were not mobile the routing tables did not change after the initial warm up period. Once the routing tables were populated, the ns-3 Static routing algorithm was able to return routes to each other node in the network.

After the warm up period, each node began generating data traffic. Data traffic was one of either UDP or TCP traffic, described below, though we performed the majority of our experiments using UDP. Each node was equipped with a Sender and a Receiver application, which resided at the topmost Application layer in the ns-3 network stack, illustrated in Figure 3.2. The Sender application was responsible for generating data and sending it into the network, and the Receiver application would listen to the network for data addressed to itself. Each node's Sender would schedule its next send event using a local exponential random variable with a mean value specified as part of the experiment. When these events were triggered, the node would uniformly randomly select another node in the network and send a fixed sized chunk of data to its respective Receiver. Our experiments were with a fixed data size and varying mean time between send events. In this way, we were able to gradually increase the offered load per node to the network and measure the effect.

As each packet was generated and sent into the network it was tagged with a monotonically increasing 32 bit sequence number unique to the originating Sender.[1] At the Receiver, the tuple formed by the originating node and the sequence number

[1]Each 16 node experiment typically generated a number of packets on the order of millions, so we are not concerned about the sequence numbers wrapping.

formed a unique packet identifier which we used to eliminate duplicates. Each Receiver maintained a buffer of the last 60 seconds worth of received packets, which was checked for duplicates before a newly arrived packet was counted. In this way, we eliminated duplicate packets at the Receiver.

At the end of each experiment we had a 20 second cool down period during which nodes would continue to pass traffic across the network but would not generate any more new traffic. In this way, we ensured that the last few packets generated had enough time to arrive at their final destination. Any packets still in the network at simulation end time after the cool down period were counted and considered lost.

5.3.1 UDP Data

For the UDP data, packet size was fixed at 1500 B. Thus, at each send event a node would send a single 1500 B packet into the network to another randomly selected node. After sending a single packet, a node would schedule its next send event, which would be sent to another randomly selected node.

1500 B packets were used because this is the MTU for Ethernet basic frames [50], and therefore seems like a reasonable size packet to work with. After IP, MAC and Phy headers, each 1500 B packet occupied 1569 B in the physical channel.

5.3.2 TCP Data

For TCP data flows, we used larger data sizes and correspondingly larger mean time between send events. When using TCP, each node would randomly select another node in the network and send it 100 kB of data. After initiating a data transfer nodes would then schedule the next sending event, which would initiate data transfer to another node in the network. Thus, it was possible that a node may have had more than one TCP transfer in progress simultaneously. Our selection of 100 kB for our data transfer size was based on the size of the average basic web page, which is on the order of a few hundred kilobytes [13]. By default, the ns-3 TCP implementation uses the New Reno implementation [15].

In this work, we tested TCP data transfers only in the grid configuration.

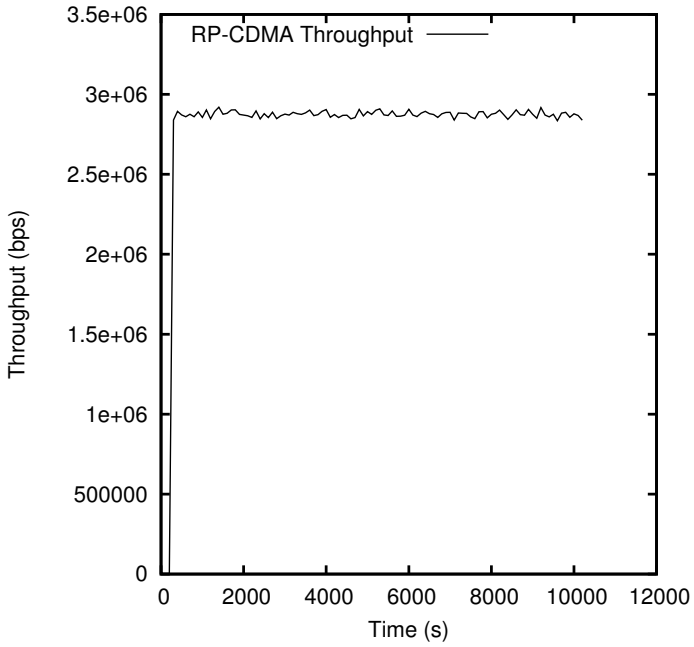


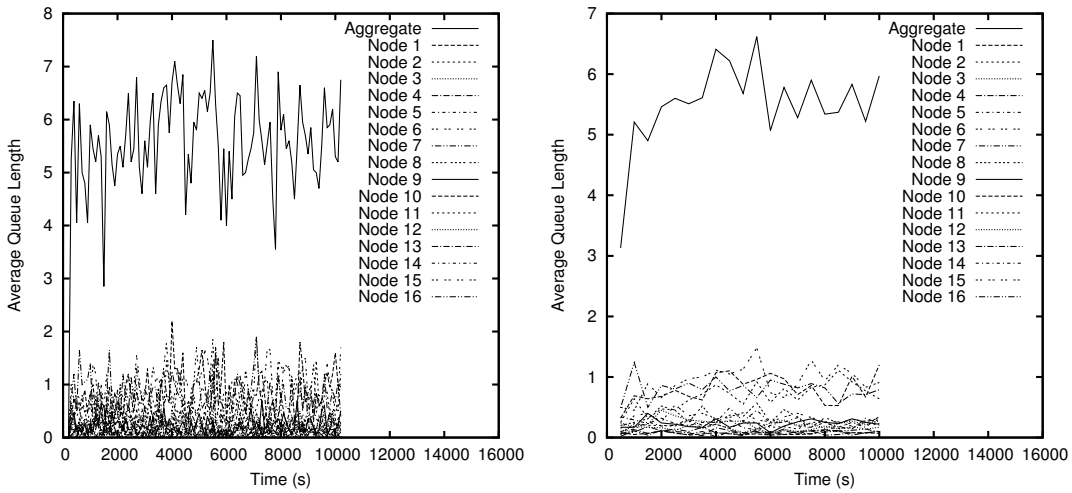
Figure 5.4: Steady state system throughput, sampled every 5 seconds and averaged over 100 second intervals. We see here that the system enters steady state almost immediately.

5.4 Calibration and Runtime

Before embarking on a series of experiments, it is prudent to evaluate our system performance at steady state and verify both that it reaches steady state and how long it takes to do so. The simplest way to do this is to periodically sample the system performance and visually identify the steady state region.

We show such a sample in Figure 5.4, which shows the sampled system throughput for a data run with 16 nodes in the grid topology and an offered load per node of approximately 0.18 Mbps, for a total offered load to the system of approximately 2.8 Mbps. In this chart, data traffic begins at 200 seconds and terminates at 10200 seconds. We see that during this period the system throughput is approximately 2.8 Mbps, with some oscillation around this value.

For the same experiment, we can also measure the data queue lengths of all the nodes in the system, which we show in Figure 5.5. We can see in this figure that the individual node queues are typically holding around one packet or less, and the system as a whole averages a little less than 6 packets waiting at any time. We can



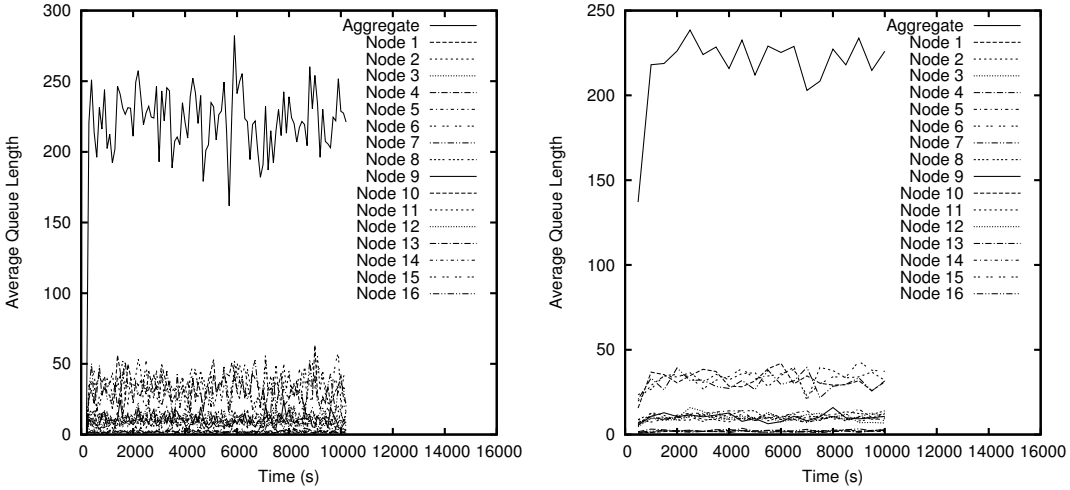
(a) Average over 100s

(b) Average over 500s

Figure 5.5: Steady State queue length. We show the average queue length of each node along with the system aggregate queue length for both (a) 100 second averages and (b) 500 second averages, sampled every 5 seconds.

increase the number of samples in each data point to eliminate some of the noise from Figure 5.5a, which results in Figure 5.5b. In this figure, we can see that only four nodes are typically holding a packet at any given time: Nodes 6, 7, 10 and 11. If we recall the grid topology from Figure 5.1, we see that these four nodes are in the middle of the grid, and therefore have more neighbours and are more likely to forward traffic across the network.

Finally, we seek to confirm that our system is stable under heavy loads, and so perform the same experiment as above but we increase the offered load per node from 0.18 Mbps to 1.0 Mbps, which is the highest offered load per node tested in any of our experiments in this work, and is also equal to the channel speed. Figure 5.6 shows our results, where we can see that the total number of packets waiting in the system oscillates between 200 and 250 for the duration of our experiment. We also see the separation of nodes into three distinct groups, which correspond to the number of neighbours each has. One set of nodes, Nodes 6, 7, 10 and 11, typically hold between 30 and 40 packets, which we again attribute to their being in the middle of the grid topology and each having four neighbours. Nodes 1, 4, 13 and 16 average around 1 or 2 packets when sampled, which we can attribute



(a) Average over 100s

(b) Average over 500s

Figure 5.6: Steady State queue length with high load. We see that the aggregate system queue length remains stable over the entire experiment, even when each node offers a load of 1 Mbps to the system.

to their being in the corners of the network, having only two neighbours each, and thus not typically forwarding as much traffic at any given time. The remaining nodes typically hold around 10 packets, which corresponds to these nodes being situated along the edges of the network and having 3 neighbours each. Critically, we see that even under heavy loads, the system reaches the steady state quickly and remains there for the duration of each experiment.

We conclude from Figures 5.4 through 5.6 that the transient in our network is fairly short, and so we do not remove it from our data. Rather, we simply run the system for a long time and use all of the data, which should effectively reduce the transient period data to insignificance. We therefore performed all of our experiments over a period of 10000 seconds, which allows us to capture significant data at the system steady state.

We have now described all of our fixed system parameters and their values, which are summarized in Table 5.2.

Fixed Parameters	
Parameter	**Value**
RNG Seed	12345
RNG Substream	1-10
Channel Speed	1 Mbps
Number of Nodes	16
Runtime	10000 s
UDP Data Size	1500 B
TCP Data Size	100 kB

Table 5.2: Fixed System Parameters

5.5 Measurements

Since our goal is to evaluate the reliability and performance of the RP-CDMA based ad hoc network, we measured the system packet loss, end to end delay, and throughput for each experiment. These three metrics together allow us to estimate the system reliability through packet loss and delay, and performance through system throughput. In the remainder of this section, we discuss our measured metrics in detail.

5.5.1 Packet Loss

Packet loss was measured by counting the number of unique packets that arrived at their destination Receiver application compared to the number of packets sent into the network by the Sender applications. Thus, packet generation and reception was counted at the application layer, which is the topmost layer in the ns-3 network stack shown in Figure 3.2. At this level, the Sender and Receiver applications are shielded from what is happening below them in the stack, and only interact with the transport layer below them. In particular, the Receiver only ever sees a packet when that packet is addressed to it specifically. Thus, intermediate nodes which merely forward packets on their way to their final destination do not see or count packets as they are forwarded along a route with multiple hops.

One of the consequences of our measurement methodology is that the actual traffic present on the network is somewhat higher than what we measure. Since we

only measure end to end packet loss at the application layer, we ignore all of the other network traffic required in order to support the network itself. Specifically, none of routing packets, retransmissions nor acknowledgement packets are seen at the application layer, since these are generated in the network and link layers. These additional sources of network traffic are present in all of our experiments, and do have an effect on our results, but are just not counted directly. Recall that OLSR continues to broadcast routing information throughout each experiment, and if we utilize an acknowledgement mechanism then acknowledgement packets and retransmissions also add to the load on the network. All of these things are present in the simulated wireless channel and therefore impact the system performance, but are not seen by the application layer and so are not counted when we measure packet loss.

We do this because we are interested in the reliability of the network from the point of view of the primary customer, which is the user. To count routing traffic and particularly retransmissions towards system packet loss is not meaningful because it does not capture the level of service afforded to the user by the network. Similarly, performing experiments in the absence of this traffic detracts from the realism of the simulation.

5.5.2 System Throughput

We calculated system throughput as a function of packet loss. Specifically, the product of the number of packets which were not lost and the size of those data packets gives us the number of bytes passed through the network, which we divide by the simulation run time to get the throughput. Again, we make these measurements from the application layer, and so only the size of the actual data portion of each packet was counted. So, if a Sender application passes a 1500 B packet down to the UDP transport layer, when that packet is eventually received at the target Receiver application, the contribution to system throughput is 1500 B.

Again, this shields our measurements from the implementation details of the network. A 1500 B packet generates 1569 B in the RP-CDMA channel, which proportionally increases the load on the network. Since we are interested in the system

performance from the point of view of the user, we only measure how much user data successfully traverses the network. As the system itself is actually transporting somewhat more than just the 1500 B sent from the application layer once we account for IP, MAC, Link, and Phy headers, some readers may wish to call our measurement the Goodput, but we use the term Throughput in our work.

5.5.3 Average Delay

End to end delay is simply the difference between the time when a packet is delivered at its final destination and when it was generated and sent into the network. We use a ns-3 packet tag to indicate the time when a packet was generated at the Sender application. This tag is checked by the destination Receiver, which records the time taken for the given packet to traverse the network. Packet Tags are a convenience abstraction in the ns-3 simulator, and do not contribute to the size of a data packet as it passes through the network stack.

5.6 2^k Factorial Analysis

Having discussed our fixed parameters and our measured values, we now turn our attention to the variable parameters listed in Table 5.1. The first three of these parameters - network topology, offered load per node and acknowledgement policy - represent the major factors in our performance study. In our results, we will first investigate system performance on the grid topology, then on random topologies, and on each topology we will investigate the effect of our acknowledgement policies. Each of our results will be presented as a set of curves showing the effect on our three measured values as a function of offered load per node. This leaves several of our RP-CDMA device parameters to investigate: the time before retransmission ($Acktime$), simultaneous packet spacing maximum (I), initial backoff maximum (B), MAC data queue length (M), and multiuser detector capability (K). To get an appreciation for how these device variables impact system performance, we can design a 2^k factorial series of experiments [20] with a fixed offered load per node, network topology, and acknowledgement policy.

We performed a 2^5 set of experiments on our grid topology, with offered load per node of 0.5 Mbps and the Eventual Ack policy. We varied the RP-CDMA device parameters $Acktime$, I, B, M, and K in the following ranges:

Variable	**Low (-1)**	**High (+1)**	**Input Label**
Acktime	0.5 s	3.0 s	x_A
I	2	10	x_B
B	2	80	x_C
M	50	∞	x_D
K	11	∞	x_E

With 5 variables, we must perform $2^5 = 32$ experiments for each of the possible combinations of the high and low values for our parameters. Table 5.3 shows each of our experimental inputs, along with the results of our 3 measured values.

We can calculate the Sum of Squares Total (SST), or total variation, for each of our measured values using the usual formula $SST = \sum_{i=1}^{2^5}(y_i - \bar{y})$, where y_i iterates over each of the results for a measured value, and $\bar{y}$ is the average of those results. We can then calculate how much of the variation is attributable to the variables $x_{A...B}$, and all of their combinations, x_{AB}, x_{AC}, ..., x_{ABCDE}. These results are shown in Table 5.4. In this table, we see that the capability of the multiuser detector (x_E) is the dominant factor which affects all three of our performance metrics. In particular, packet loss and system throughput are almost entirely attributable to the capability of the multiuser detector, which accounts for approximately 98% of the variation in our results. End to end delay performance is also mostly attributable to the capability of the multiuser detector, $x_E \approx 29\%$, though the size of the MAC data queue also has a significant impact, $x_D \approx 15\%$, as well as the interaction between the two, $x_{DE} \approx 15\%$. The effects of the simultaneous packet spacing, x_B, and the initial backoff window, x_C, both also accounted for a non-negligible amount of the variation in the end to end delay, accounting for $x_B \approx 2.9\%$ and $x_C \approx 6.3\%$ respectively.

x_A	x_B	x_C	x_D	x_E	**Loss (%)**	**Throughput (Mbps)**	**Delay (ms)**
1	1	1	1	1	0.3393	7.976	205.0
1	1	1	1	-1	66.56	2.676	8963
1	1	1	-1	1	2.086	7.836	205.2
1	1	1	-1	-1	57.89	3.369	817.4
1	1	-1	1	1	0.3113	7.977	141.2
1	1	-1	1	-1	72.33	2.214	3211
1	1	-1	-1	1	0.7741	7.940	140.4
1	1	-1	-1	-1	68.65	2.508	1344
1	-1	1	1	1	0.2783	7.982	165.6
1	-1	1	1	-1	66.31	2.696	5230
1	-1	1	-1	1	1.144	7.912	166.2
1	-1	1	-1	-1	59.44	3.246	794.7
1	-1	-1	1	1	1.850	7.854	159.7
1	-1	-1	1	-1	69.23	2.461	1368
1	-1	-1	-1	1	2.386	7.811	156.0
1	-1	-1	-1	-1	68.68	2.505	1211
-1	1	1	1	1	0.3409	7.976	207.1
-1	1	1	1	-1	69.08	2.474	10582
-1	1	1	-1	1	2.209	7.826	207.0
-1	1	1	-1	-1	57.74	3.381	793.7
-1	1	-1	1	1	0.3445	7.975	141.8
-1	1	-1	1	-1	74.77	2.018	3295
-1	1	-1	-1	1	0.5354	7.959	136.2
-1	1	-1	-1	-1	69.08	2.474	949.8
-1	-1	1	1	1	0.2981	7.980	166.2
-1	-1	1	1	-1	67.10	2.632	5228
-1	-1	1	-1	1	1.244	7.904	165.8
-1	-1	1	-1	-1	59.82	3.215	742.0
-1	-1	-1	1	1	1.650	7.870	158.1
-1	-1	-1	1	-1	69.17	2.467	954.6
-1	-1	-1	-1	1	2.370	7.813	154.6
-1	-1	-1	-1	-1	68.46	2.523	767.4
			Average		33.83	5.295	1529

Table 5.3: Loss, Throughput and Delay results for 2^k analysis.

	Contrasts q_x			Variation & Percent Explained					
				Loss		Throughput		Delay	
Parameter	**Loss**	**Throughput**	**Delay**	**Variation**	**% Explained**	**Variation**	**% Explained**	**Variation**	**% Explained**
x_A	-0.1859	0.0149	-11.5	1.105	0.003	0.0071	0.003	4272	0.002
x_B	0.1123	-0.0091	429.7	0.4039	0.001	0.0027	0.001	5910542	2.899
x_C	-1.83	0.1472	635.9	107.7	0.311	0.6937	0.312	12941453	6.347
x_D	1.170	-0.0937	982.1	43.81	0.126	0.2807	0.126	30867368	**15.14**
x_E	-32.6	2.616	-1361	34206	**98.84**	219.1	**98.84**	59350045	**29.11**
x_{AB}	-0.1371	0.0110	-68.7	0.6011	0.002	0.0038	0.002	151468	0.074
x_{AC}	-0.0507	0.0041	-84.9	0.0823	0	0.0005	0	231015	0.113
x_{AD}	-0.1608	0.0129	-68.9	0.8273	0.002	0.0053	0.002	152234	0.074
x_{AE}	0.1970	-0.0158	11.71	1.242	0.003	0.0080	0.003	4394	0.002
x_{BC}	-0.0741	0.0058	152.8	0.1758	0	0.0011	0	747572	0.366
x_{BD}	0.3987	-0.0319	402.5	5.086	0.014	0.0326	0.014	5184532	2.543
x_{BE}	-0.3800	0.0304	-424	4.621	0.013	0.0295	0.013	5753879	2.822
x_{CD}	0.6254	-0.0501	696.4	12.51	0.036	0.0802	0.036	15519271	7.612
x_{CE}	1.692	-0.1352	-617	91.61	0.264	0.5851	0.264	12189015	5.978
x_{DE}	-1.62	0.1304	-981	84.89	0.245	0.5438	0.245	30815154	**15.11**
x_{ABC}	0.0616	-0.0049	-34.6	0.1213	0	0.0008	0	38389	0.018
x_{ABD}	-0.1421	0.0114	-63.9	0.6460	0.002	0.0041	0.002	130873	0.064
x_{ABE}	0.1361	-0.0109	68.62	0.5926	0.002	0.0038	0.002	150679	0.073
x_{ACD}	-0.0194	0.0015	-36.9	0.0120	0	0	0	43649	0.021
x_{ACE}	0.0090	-0.0007	84.29	0.0026	0	0	0	227369	0.111
x_{ADE}	0.1678	-0.0134	68.59	0.9014	0.002	0.0058	0.002	150584	0.073
x_{BCD}	-0.1462	0.0117	160.7	0.6842	0.002	0.0044	0.002	827201	0.405
x_{BCE}	0.5931	-0.0475	-138	11.25	0.032	0.0723	0.032	614093	0.301
x_{BDE}	-0.4738	0.0379	-402	7.183	0.020	0.0460	0.020	5185835	2.543
x_{CDE}	-0.8454	0.0677	-697	22.87	0.066	0.1465	0.066	15558394	7.631
x_{ABCD}	0.0028	-0.0002	-35.5	0.0003	0	0	0	40542	0.019
x_{ABCE}	-0.0611	0.0049	34.36	0.1195	0	0.0008	0	37799	0.018
x_{ABDE}	0.1162	-0.0093	63.68	0.4319	0.001	0.0028	0.001	129764	0.063
x_{ACDE}	0.0375	-0.0030	37.14	0.0449	0	0.0003	0	44140	0.021
x_{BCDE}	-0.0040	0.0003	-160	0.0005	0	0	0	826589	0.405
x_{ABCDE}	0.0281	-0.0023	35.94	0.0254	0	0.0002	0	41354	0.020

Table 5.4: Percent variation explained by 2^k factors

From this analysis, we expect that for a given load the dominant factor impacting packet loss and system throughout on the grid topology will be the capability of the multiuser detector, with the other variables making almost no difference. This is encouraging for our thesis that the use of a multiuser detector in RP-CDMA can result in significant reliability and performance benefits in our wireless ad hoc network.

5.7 Summary

In this chapter we have described our experimental setup, including the simulator configuration, network topology, node behaviour, experimental transient, steady state and runtime. We have also described our system measurements, which we will use to evaluate system performance in our experiments, and performed a basic analysis of how our variable system parameters impact system performance on our grid topology. We now move on to our results.

Chapter 6

Results

In this chapter, we present our results. Along with the algorithms which constitute our MAC and acknowledgement policies, this performance study is the main contribution of of our work. Here, we will explore the reliability and performance of our RP-CDMA network device on both grid and random topologies, and investigate the effectiveness of our acknowledgement policies. After an initial assessment and comparison to 802.11 CSMA, we will investigate the effect of changing our variable device parameters with the aim of maximizing system reliability and performance.

Each of the results in this chapter shows the effect of varying one of the variable parameters from Table 5.1 while keeping the others fixed. For each value of our varied parameter we generated a data series measuring the system performance as the offered load per node was increased. All of our data series were then plotted on the charts, which allows us to see the effect of changing the variable parameter. Each data series was composed from 10 independent trials with identical inputs but incremented RNG substream. Error was calculated as the 95% confidence intervals over our 10 independent trials.

This chapter proceeds through our results starting with our grid topology in Section 6.1. On this topology, we initially compare 802.11 performance with our RP-CDMA device without acknowledgements. We then investigate the effect of varying the capability of the multiuser detector, K, the effect of the Immediate and Eventual acknowledgement mechanisms, and also briefly investigate TCP performance. With this initial investigation on grid topologies completed, we present our results on random topologies in Section 6.2. On random topologies we investigated

the effect of each of our variable system parameters, with the aim of maximizing the RP-CDMA system performance. After analysing the effects of each of our system parameters, we will again compare the RP-CDMA result to that of 802.11.

6.1 Grid Topology

Recall that our grid topology consisted of 16 nodes separated such that they could communicate with their neighbours on the x and y axes, but not diagonally, as per Figure 5.1. We present results here for 802.11, RP-CDMA without acknowledgements, with the Immediate Ack Policy, and finally with the Eventual Ack policy. Our aim in this series of experiments is to analyse the relative performance of our system and acknowledgement policies in a fixed, regular topology. We then apply the lessons learned on the grid system to random topologies in Section 6.2.

6.1.1 802.11 CSMA and RP-CDMA No Ack

We begin with a simple evaluation of the 802.11 system and compare it to RP-CDMA without acknowledgements. Before presenting our results, we briefly discuss how we fairly compare 802.11 to RP-CDMA .

Our experiments with 802.11 were performed simply by replacing the RP-CDMA network device with the stock ns-3 802.11 network device in the simulator and running the same series of experiments that we performed with RP-CDMA . In order to compare the two fairly, we limited the RP-CDMA experiments to the same bandwidth as that occupied by 802.11.

In the 1 Mb DSSS mode, 802.11 utilizes differential binary phase shift keying (DBPSK) to transmit data in the 2.4 GHz ISM band. Each symbol is DSSS modulated with an 11-chip Barker sequence, which spreads the 1 Mbps data signal over 22 MHz using the formula $B = 2R\|\vec{c}\|$, where R is the data rate, and $\|\vec{c}\|$ is the magnitude of the spreading code used [51, 48]. The bandwidth occupied by 802.11 is therefore 22 MHz [48].

We can construct a corresponding case with RP-CDMA by utilizing the same data rate and spreading code length. With a spreading code length of 11, we as-

sume we can have a multiuser detector with capability at least $K = 11$ [42]. With equal bandwidth, we can fairly compare the performance of the 802.11 CSMA protocol with that of our RP-CDMA protocol. In addition to constraining RP-CDMA to a multiuser detector with $K = 11$ when comparing to 802.11, we also idealized the 802.11 Phy to ignore interference effects. Because our experiments with RP-CDMA were conducted under the assumption that the multiuser detector can successfully resolve up to K overlapping signals, and therefore ignored interference effects in the channel, we made the same assumption when experimenting with 802.11.

With interference effects ignored, we can isolate the relative effectiveness of the 802.11 CSMA collision avoidance approach and contrast it with the RP-CDMA approach of mitigating collisions by putting packets into individual channels. We disabled the optional 802.11 RTS/CTS mechanism throughout, since it does not obviously improve system performance, may actually impair it [55, 40], and our initial experiments indicated it was of limited usefulness.

Figure 6.1 shows our results for No Ack on the grid topology, where we have compared to the 802.11 result under the same configuration. Recall that in No Ack, nodes began transmitting packets after a short random backoff period when they were not busy receiving any packets and continued to send packets simultaneously until there were no more to send. Receiving nodes did not send acknowledgement packets, and so if a packet was lost in the network then the system did not try to recover. In these results, we limited our multiuser detector to capability $K = 11$, in order to fairly compare 802.11 with RP-CDMA .

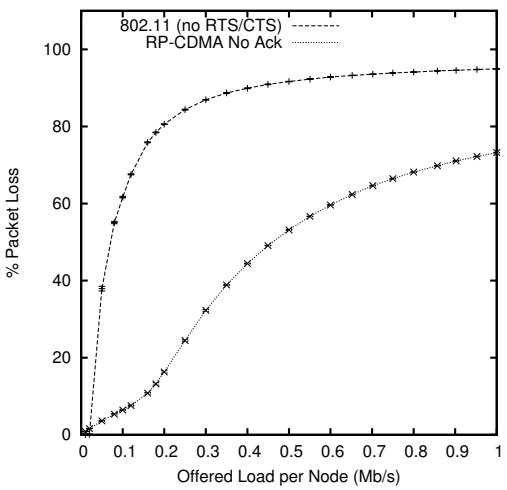
We see that even when limited to $K = 11$, RP-CDMA offered superior performance across the entire tested range. We can see in Figure 6.1a that RP-CDMA lost fewer packets than 802.11 under the same loads, which corresponds to the higher measured system throughput shown in Figure 6.1c. We see in Figure 6.1b that RP-CDMA offered much improved end to end delay compared to 802.11 and was on the order of 200 ms at the highest load tested. The end to end delay result for 802.11 has a strange artifact around 0.1 Mbps where delay briefly decreases, then resumes increasing as as offered load per node reaches approximately 0.1 Mbps.

This is a byproduct of the grid network topology, and indicates that as the load on the system increases its multihop capability is affected. Specifically, as the 802.11 system drops more packets, the only packets that successfully reach their intended Receiver - and which are subsequently measured - are those which do not require a large number of hops. Thus, the measured average delay briefly decreases, and then steadily rises again as load continues to increase.

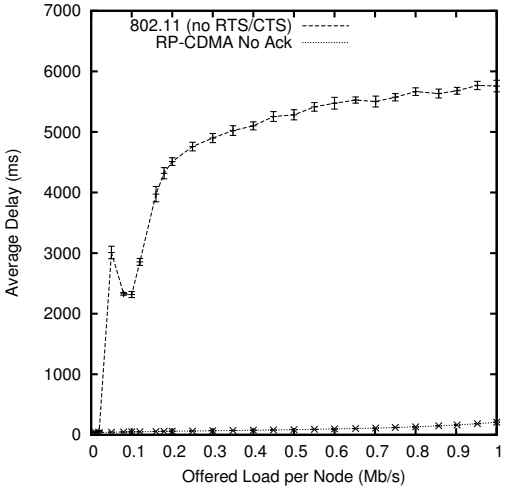
We can examine the relative causes for packet loss in the RP-CDMA case to try to determine why packets were lost in the network. For the RP-CDMA No Ack experiments, we have three possible causes for packets being lost at the receiver: Header Collisions, Transmitting, and Receiving errors. A Header Collision happens if more than one packet header overlaps at the receiver. In this case, the receiver will only synchronize to the first packet to arrive and all others will be lost.[1] A Transmitting error occurs when a packet arrives while the intended receiver is transmitting, and is thus incapable of receiving due to its half-duplex radio. All packets that arrive while a node is transmitting are dropped. A Receiving error can occur for a number of reasons in our model, but in the case of RP-CDMA No Ack, Receiving errors occur when a particular receiver's multiuser detector capability is exceeded. So for a multiuser detector with capability $K = 11$, any packets that arrive concurrently beyond the 11th packet are lost.

Figure 6.2 shows the relative reasons for packet loss as a function of offered load per node for the RP-CDMA No Ack experiment with overall losses shown in Figure 6.1a. Figure 6.2 shows us that as we increase the offered load per node, the fraction of lost packets that are lost due to exceeded multiuser detector capability grows, while the fraction of packets lost due to header collisions drops. It makes sense that as the offered load per node increases that more packets would be lost due to exceeded multiuser detector capability, and thus proportionally less due to

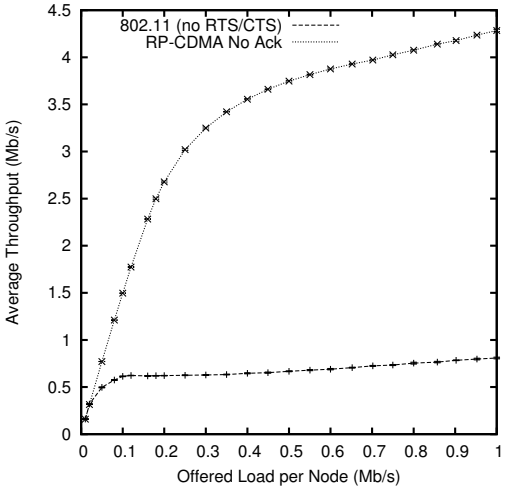
[1]This is exactly the same behaviour as the ns-3 802.11 implementation, which continues to attempt to decode the first packet to arrive and decides afterwards whether the interference from the collided packet is sufficient to cause the first packet to be lost. Some models would drop all packets involved in a collision [17], but in the ns-3 802.11 model and in reality the receiver would continue to attempt to decode the first packet. The possibility that some data may be recoverable from CDMA systems even in the event of collisions is the same property which allows Spread ALOHA using a single spreading code to recover from some collisions [4].



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Grid
Ack Policy	No Ack
Acktime	N/A
I	10
B	10
M	∞
K	11

(d) Parameters

Figure 6.1: RP-CDMA with no acknowledgements (No Ack), and limited to multiuser detector capability $K = 11$ on the grid topology. Without any acknowledgement mechanism, RP-CDMA offers superior metrics across the entire tested range compared to 802.11.

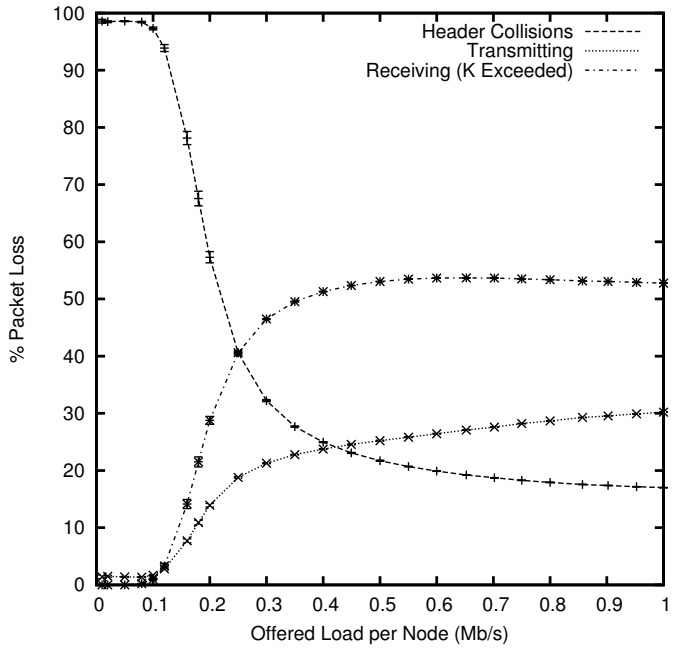
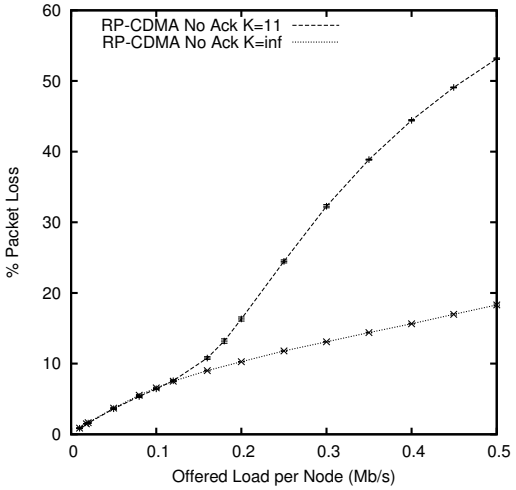


Figure 6.2: RP-CDMA No Ack Loss Reasons with $K = 11$. We see that as the offered load per node increases, the proportion of packets that are lost due to the capability of the multiuser detector being exceeded grows as a fraction of all lost packets.

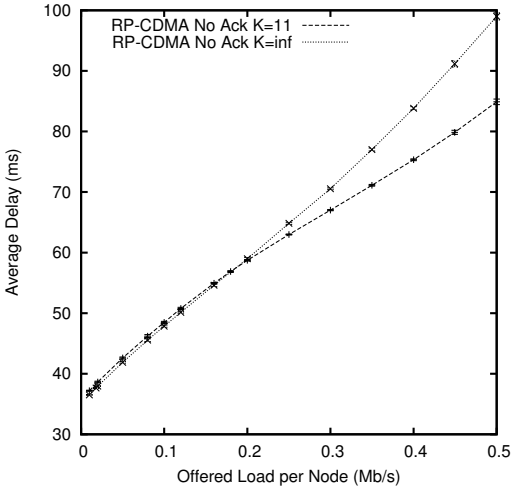
packet header collisions. It is thus worthwhile to examine how system performance could be affected by a more capable multiuser detector.

Figure 6.3 shows our results when we remove the $K = 11$ restriction on our RP-CDMA multiuser detector, keeping all other things constant. Compared to the $K = 11$ case, increasing the capability of the multiuser detector results in lower packet loss and higher throughput at the cost of marginally higher end to end delay. These results are expected, as increasing the number of delivered packets in the network is expected to increase the time required to deliver each of them. Our result for packet loss is particularly encouraging, as the number of packets lost drops from more than 50% to less than 20% at the upper end of our offered load.

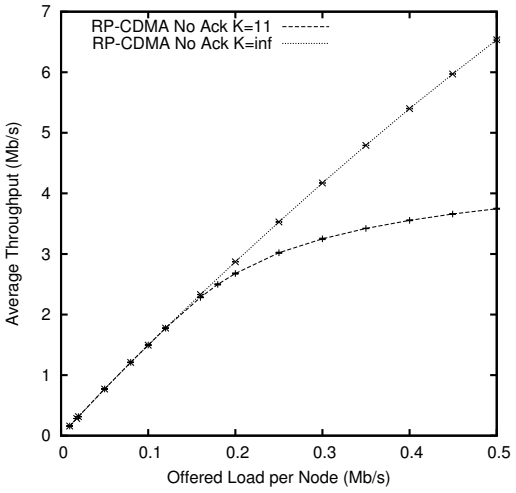
We have now seen that with equal bandwidth, our RP-CDMA based system outperforms the 802.11 CSMA system, and that given an infinite multiuser detector, RP-CDMA can perform much better. Since we do not have infinite capability multiuser detectors in reality, it makes sense to examine what multiuser detector capability would be required to approach the $K = \infty$ ideal case.



(a) Average Packet Loss



(b) Average Delay

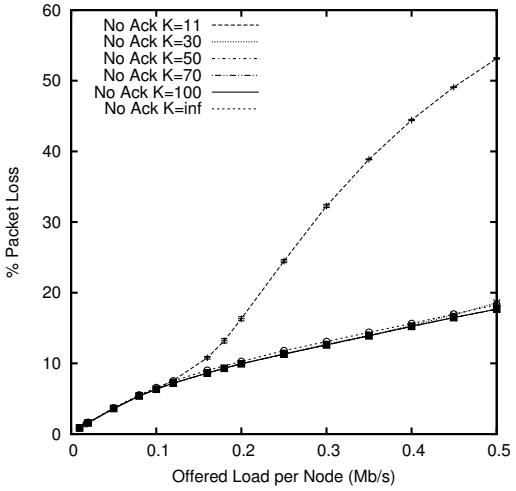


(c) Average System Throughput

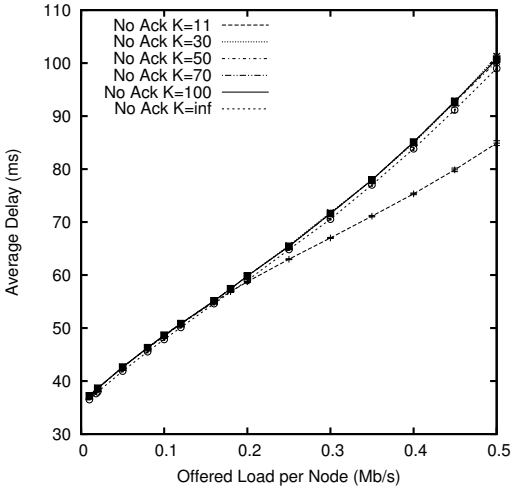
Topology	Grid
Ack Policy	No Ack
Acktime	N/A
I	10
B	10
M	∞
K	Varied

(d) Parameters

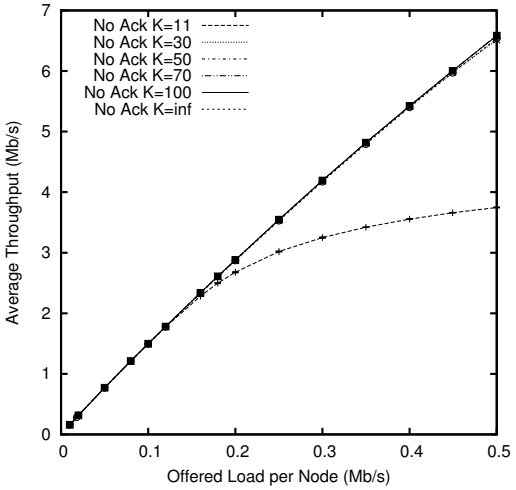
Figure 6.3: RP-CDMA No Ack with $K = \infty$. We see that increasing the capability of the multiuser detector, K, from 11 to ∞ yields superior performance in all metrics.



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Grid
Ack Policy	No Ack
Acktime	N/A
I	10
B	10
M	∞
K	Varied

(d) Parameters

Figure 6.4: RP-CDMA No Ack for varying multiuser detector capability, K. We see here that for values of K greater than approximately $K = 50$ that our results are almost identical to the infinitely capable multiuser detector.

Figure 6.4 shows our results for a range of values for K. We see that performance approaches the ideal result as K increases, with $K = 50$ being almost indistinguishable from the ideal, $K = \infty$, result. For values of K higher than 50, the results overlap with the $K = \infty$ result. Our review of the literature leads us to believe that detector capabilities in the range of $K = 50$ are not unreasonable [36, 17].

While it is encouraging that our RP-CDMA based device can significantly outperform the 802.11 CSMA based device, we still lose significant numbers of packets under relatively high loads, even with infinitely capable detectors. This observation motivates our investigation of acknowledgement mechanisms, which we begin in the following section.

6.1.2 Immediate Ack

Recall that the Immediate Ack policy is a fixed time interval acknowledgement policy, and so a receiving node will guarantee that an acknowledgement will be sent for a received packet within some fixed time interval. Because the sending node knows when to expect an acknowledgement, it will stop transmitting and wait to receive acknowledgements after sending some number of simultaneous packets.

Immediate Ack is compared to No Ack in Figure 6.5, which shows Immediate Ack with various values for *Acktime* plotted against No Ack for the ideal, $K = \infty$ case. We see that under light load the Immediate Ack policy can achieve lower end to end packet loss with roughly equal throughput compared to No Ack, but at the cost of higher latency as nodes must stop and wait for acknowledgements before continuing. As the offered load per node increases, the Immediate Ack policy begins to fail and packet losses climb rapidly. Under higher loads, the Immediate Ack policy regularly breaks off packet receptions in order to send acknowledgements and retransmissions. This has a cascading effect, where cancelled receptions incur more retransmissions, which cause more cancelled receptions, and so on. Under these load conditions it becomes better to simply not send acknowledgements at all, and we see that the Immediate Ack policy eventually begins to perform worse in every metric than No Ack. Note the scale of the horizontal axis in Figure 6.5,

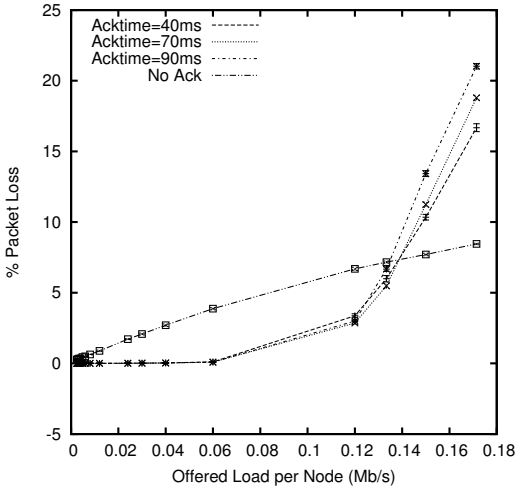
which is shorter than the scale we have been using previously.

We attempted several modifications to the Immediate Ack protocol, including varying values for *Acktime* in the receiving node and varying values for the acknowledgement transmission time in the receiving node. None of our experiments yielded better results than those presented in Figure 6.5. The fundamental problem with the Immediate Ack policy is that nodes must continually switch between sending and receiving in order to meet the fixed times for sending and receiving acknowledgement packets, which results in compounding packet losses as the load on the network increases. On the receiving side packets are lost as the receiving node switches to transmit acknowledgements and cuts off packet reception, and on the sending size, bandwidth is wasted and delay incurred as sending nodes wait for acknowledgements.

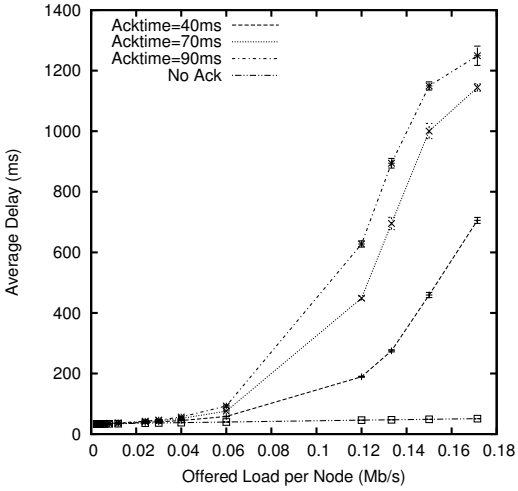
Given this result, we ask how do we maintain the desirable characteristics of the No Ack policy - the low latency and high system throughput - while keeping packet loss low or negligible, such as with the Immediate Ack policy under low load. Our proposed solution to this problem is the Eventual Ack policy.

6.1.3 Eventual Ack

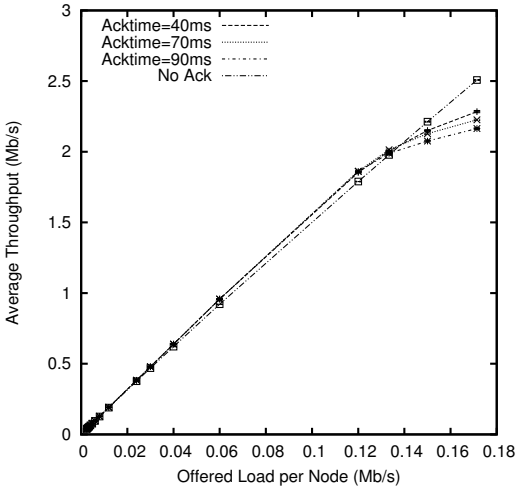
Recall that the rationale behind the Eventual Ack policy was that even under relatively high load a significant portion of packets successfully traverse the network, and we need only an effective way to identify those few lost packets. With Eventual Ack, the sending node thus continues to send data packets and does not wait for acknowledgements, just as in the No Ack policy. When the channel becomes free a receiving node sends any accumulated Acks first, then any packets which need to be retransmitted, and finally any new packets. When a receiving node eventually transmits acknowledgement packets, the corresponding sending nodes can then identify which, if any, of their previously transmitted packets were lost and schedule them for retransmission. From the point of view of the MAC layer, the Eventual Ack policy superficially resembles the No Ack policy in that acknowledgements are treated much the same as ordinary data traffic, and the sending node does not wait for acknowledgements before continuing with other transmissions. Eventual Ack differs



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Grid
Ack Policy	Immediate Ack
Acktime	Varied
I	10
B	10
M	∞
K	∞

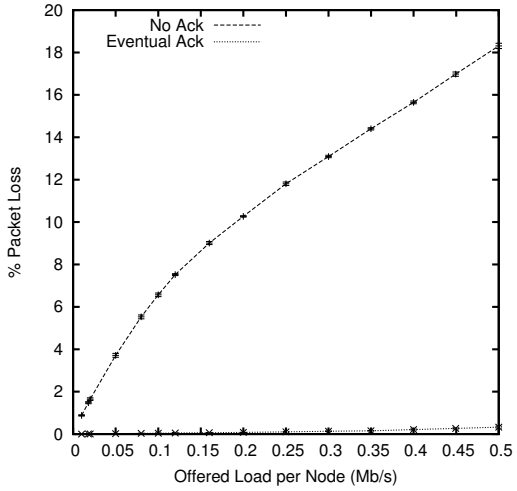
(d) Parameters

Figure 6.5: Immediate Ack performance with ideal multiuser detector and grid topology. We see that sending acknowledgements after receiving a packet can improve link reliability under light load, but these gains are lost as load increases.

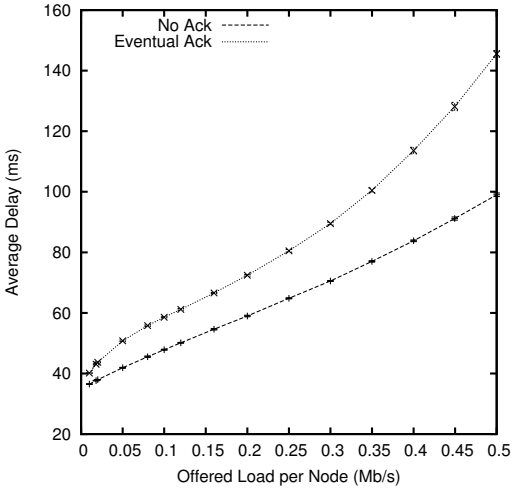
from No Ack in that it does use acknowledgements to confirm packet reception, and differs from Immediate Ack in relying primarily on the ordering of acknowledgements to infer lost packets, leaving the timeout mechanism as only a guard against all packets to a given neighbour being lost.

Our aim in this section is to determine the bounds of our approach. By testing our system on the fixed, regular grid topology, we can determine the upper bounds of our system performance by testing with idealized parameters. To this end, our experimental baseline in this section will use the same parameters as our best results for No Ack with the infinitely capable multiuser detector, namely $K = \infty$, initial backoff maximum $B = 10$, simultaneous transmission backoff $I = 10$ and data queue size $M = \infty$. Figure 6.6 shows our results comparing RP-CDMA No Ack to Eventual Ack under these conditions. These results show the effectiveness of the Eventual Ack policy, which offers superior packet delivery rates and system throughput compared to No Ack across the tested range. Our system throughput result shows that even under high load the system effectively carries all of the traffic offered to it, at the cost of a slightly higher delay. At the highest load tested, packet loss for the Eventual Ack policy was on the order of 0.3%, which indicates that our acknowledgement mechanism successfully identified and retransmitted lost packets. This result is very encouraging, as it shows that given sufficient resources, the RP-CDMA based net device with a simple acknowledgement policy is capable of achieving negligible packet loss with reasonable delay on a multihop ad hoc wireless network, for our given number of nodes and network topology.

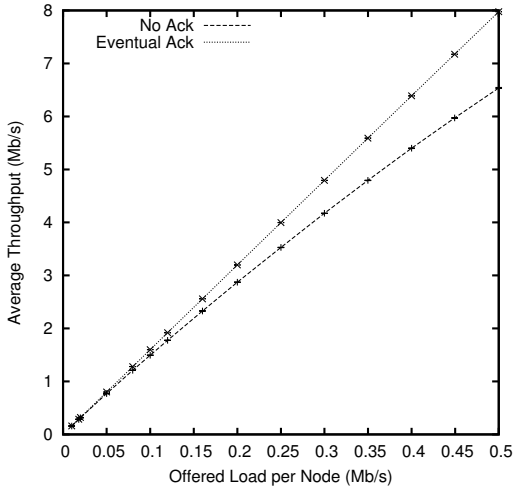
We tested a range of values for *Acktime* in the range $0.25 \geq Acktime \geq 1.5$ s and found that for values greater than 0.5 s, the results were effectively identical. We therefore take the parameters in Figure 6.6d as our baseline, and our results with these parameters on the grid topology to be the upper bound on the possible performance of our system. From here, we will attempt to approach this upper bound with different network transport and topologies. In the remainder of our results, we will first briefly investigate TCP performance on the grid, and we will then return to UDP data on random topologies.



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Grid
Ack Policy	Varied
Acktime	0.5 s
I	10
B	10
M	∞
K	∞

(d) Parameters

Figure 6.6: Eventual Ack policy compared to No Ack on the grid topology with $K = \infty$ and *Acktime*$= 0.5$ s. Eventual Ack offers lower packet loss within our tested range while maintaining low end to end delay and high throughput.

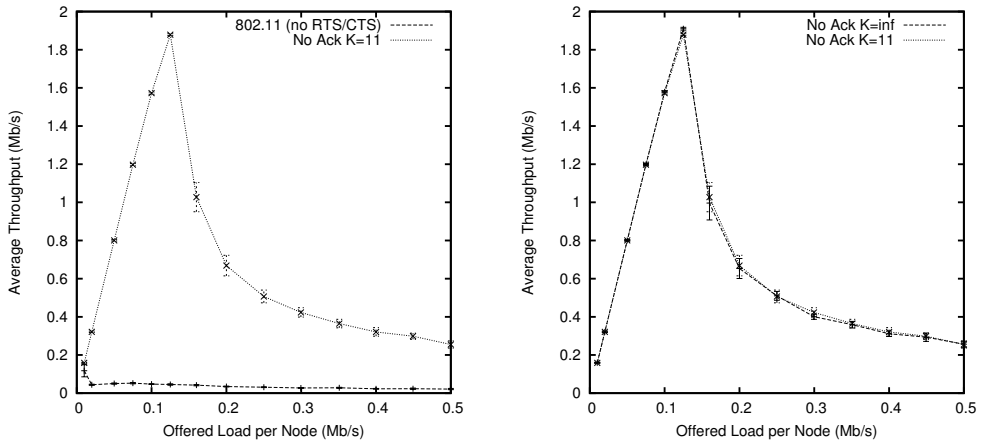
6.1.4 TCP Data

In this section, we briefly investigate TCP performance on the grid. We saw above that with UDP data, the system was able to effectively carry all of the load offered to it by the nodes. TCP performance on wireless links has previously been identified as a significant problem [9, 14, 37], and there have been several proposals in the literature to modify TCP in various ways to overcome these problems [6]. Among the reasons often cited for poor TCP performance over wireless links is that packet loss is often attributable to collisions or other link layer loss, which TCP incorrectly reacts to as congestion [14]. With a reliable RP-CDMA link, we expect that TCP performance will be improved.

We recall that with TCP data, each node would randomly select another node in the network and send it 100 kB via TCP. Because TCP breaks up this data and transfers it in smaller chunks, we show here only our system throughput result and omit the end to end delay of each TCP encapsulated packet, which is not indicative of overall performance, and the packet loss, which is not meaningful because TCP guarantees packet delivery.

We show our results with TCP data transfers in Figure 6.7, which shows (a) TCP performance with RP-CDMA compared to 802.11 and (b) TCP performance with RP-CDMA with both $K = 11$ and $K = \infty$. In these results, we see that RP-CDMA does indeed improve system throughput compared to 802.11, but does not approach the performance obtained with UDP transfers. Furthermore, we see in 6.7b that this result is not sensitive to the capability of the multiuser detector, and that increasing the detector capability from $K = 11$ to $K = \infty$ does not improve TCP performance.

It is natural to ask what is happening to cause these results. Whereas system throughput with UDP packets improved steadily as we increased the load offered to the system, TCP briefly improves, but beyond approximately 120 kbps of offered load per node rapidly declines and begins to level off at approximately 200 kbps of system throughput. We examined the congestion windows of each of the TCP sockets in the system, and found that the window size steadily increased initially, but would sometimes collapse down to the slow start value and remain there. As



(a) RP-CDMA compared to 802.11 (b) RP-CDMA $K = 11$ compared to $K = \infty$

Figure 6.7: TCP Data Transfer Results. TCP achieves better performance with RP-CDMA than with 802.11, but does not approach the performance achieved with UDP.

TCP sockets reduced their throughput, the number of sockets with data to transfer would increase as the nodes continued to initiate transmissions to other nodes. With more sockets attempting transfers, system throughput was further degraded as contention for available resources increased. We know from our results with RP-CDMA No Ack that some packets are lost in the system (Figure 6.6), and we therefore suggest that it is possible that the reasons for our TCP performance results are partially attributable to the same causes as identified with 802.11 - namely that losses at the link layer cause TCP to react inappropriately and assume network congestion.

Our experiments with TCP in ns-3 were also delayed by bugs in the ns-3 TCP implementation. Until very recently, a bug existed in the ns-3 TCP stack which caused sockets to occasionally stall completely, which made it impossible to perform meaningful experiments. While we do not believe that our results here are attributable to problems with the ns-3 TCP implementation, it is nonetheless a possibility.

Regardless, our results with TCP are somewhat disappointing. We had hoped that a reliable link would significantly alleviate or eliminate the well known problems with TCP over wireless links, but found that while RP-CDMA does perform

better than 802.11, it does not fundamentally change the nature of the problem, which is that under heavier loads TCP performance rapidly diminishes. We believe this topic deserves further investigation, which we recommend in Section 7.2, but for now return to UDP data transfers and investigate our system performance over random topologies.

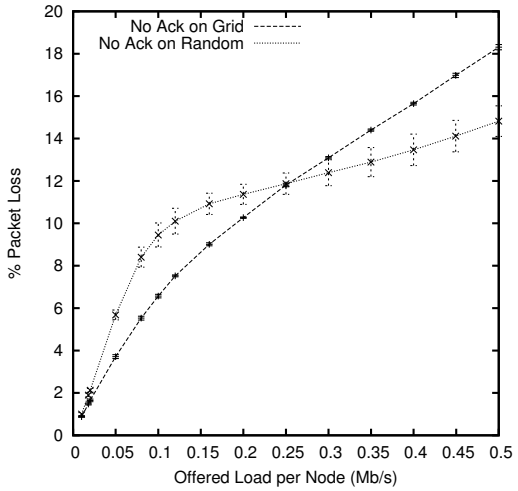
6.2 Random Topologies

Now that we have seen how our system performs on a regular grid topology, we investigate random topologies. We will use UDP data transfers in the remainder of our results, and our aim is to approach the result obtained on the grid topologies, which, with UDP data, was effectively negligible packet loss and low latency. We shall see in this section that achieving high reliability and performance on random topologies is more difficult than on the grid topology, and so we will investigate the performance effects of varying our system parameters in the following sections. Once we have identified a set of our system parameters that achieve high reliability and performance with ideal receivers, we will take away our ideal receiver assumptions and will limit both the size of the data queue, M, and the multiuser detector capability, K. With this completed, we will return to our comparison with 802.11.

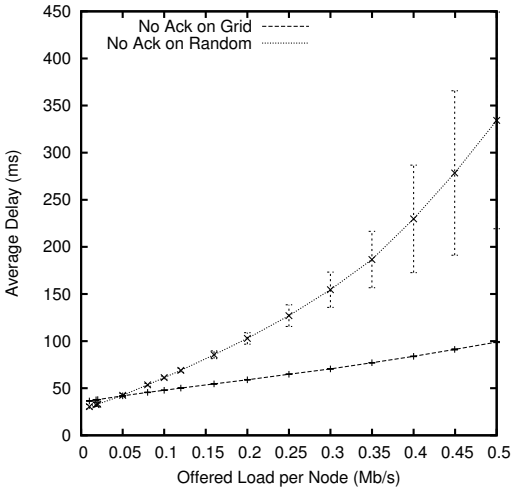
6.2.1 No Ack

We begin again with No Ack. Figure 6.8 shows a comparison between No Ack on the grid topology and No Ack on random topologies with ideal receivers. In this figure, we can see that the performance on our random topologies was similar to that of the grid topology, with the exception of end to end delay, which increased rapidly compared to the grid case.

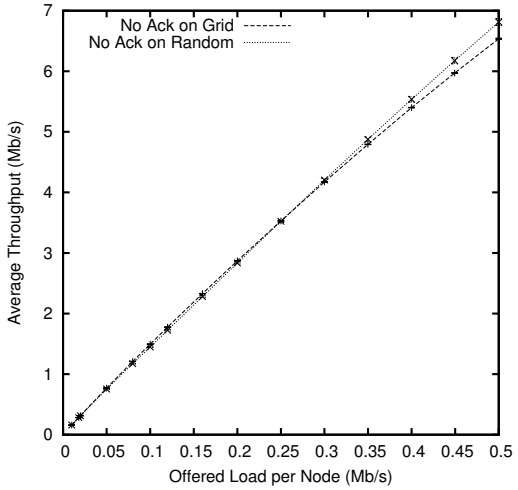
It is straightforward to understand why the average end to end delay is higher on the random topologies. Compared to the grid topology, the random topologies have areas of relatively high and low node density. In the areas of high node density, it is less probable that a given node will begin transmitting first after determining that



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Varied
Ack Policy	No Ack
Acktime	N/A
I	10
B	10
M	∞
K	∞

(d) Parameters

Figure 6.8: No Ack performance on grid and random topologies. We see that on the random topologies, packet loss is slightly less, but delay is higher.

it is safe to transmit.[2] Thus, packets waiting to be transmitted from a given sending node will wait relatively longer in the outbound data queue. As a result, the average time taken to reach a given destination is longer. Note that even with this increased delay, system throughput is comparable to the grid topology case, and packet loss is slightly lower under higher loads.

Despite this positive result, our RP-CDMA device still exhibits unacceptably high packet loss without an acknowledgement mechanism, and so we turn our attention to the Eventual Ack policy.

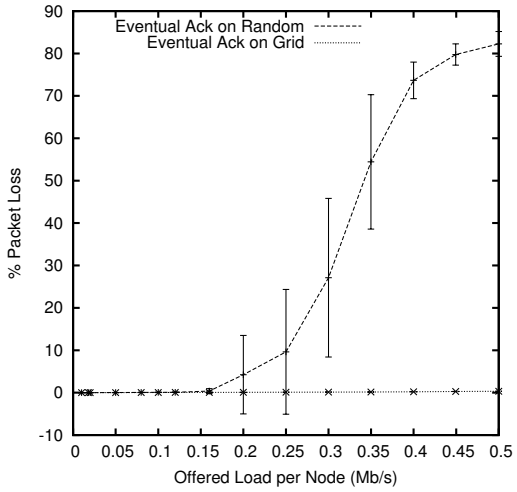
6.2.2 Eventual Ack

On random topologies, Eventual Ack performance was seriously degraded compared to the grid topology case, as can be seen in Figure 6.9. In this figure, we can see that Eventual Ack on random topologies initially performed similarly to the grid result, but as offered load per node exceeded approximately 0.15 Mbps, system performance degraded rapidly. At this highest offered load per node, packet loss on the random topologies was greater than 80% and end to end delay was on the order of 30 seconds.

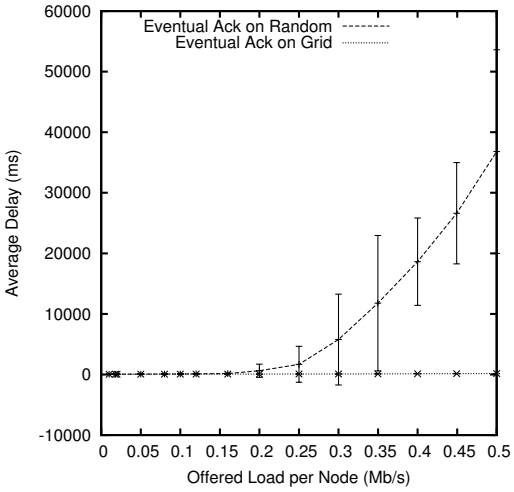
When we compared Eventual Ack to No Ack on the random topologies the results were similar and are shown in Figure 6.10. In this figure, we see that Eventual Ack initially performed better than No Ack, but as the offered load per node increased, the performance began to degrade. Again, this indicates the system sensitivity to node density. In particular, when several nodes were all within transmission range of one another each node waited longer before gaining access to the channel. As each node offered more load to the network, it became more difficult for any node to transmit their queued acknowledgements before the *Acktime* retransmission timers fired in the respective sending nodes. In this case, a sending node retransmitted prematurely, which caused further load on the network, resulting in more delays and premature retransmissions. The result was that system performance degraded rapidly beyond a particular offered load per node.

We have many parameters which we can vary in our MAC, and in the following

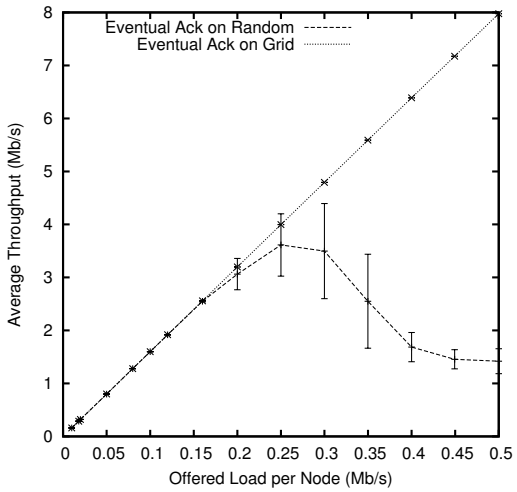
[2]The probability of a node transmitting first is shown in Appendix Equation A.14



(a) Average Packet Loss



(b) Average Delay

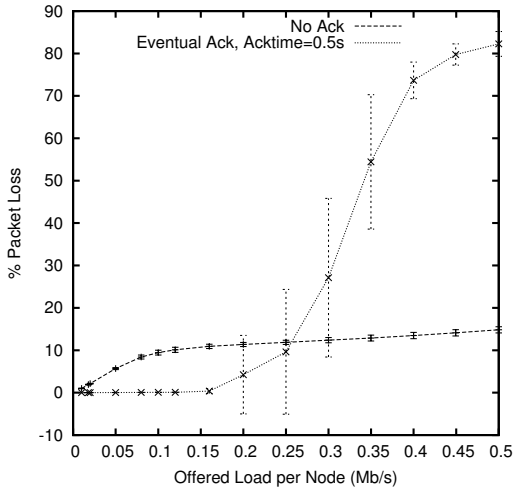


(c) Average System Throughput

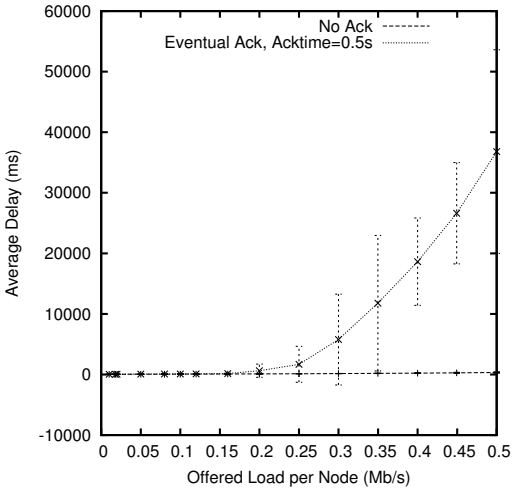
Topology	Varied
Ack Policy	Eventual Ack
Acktime	0.5 s
I	10
B	10
M	∞
K	∞

(d) Parameters

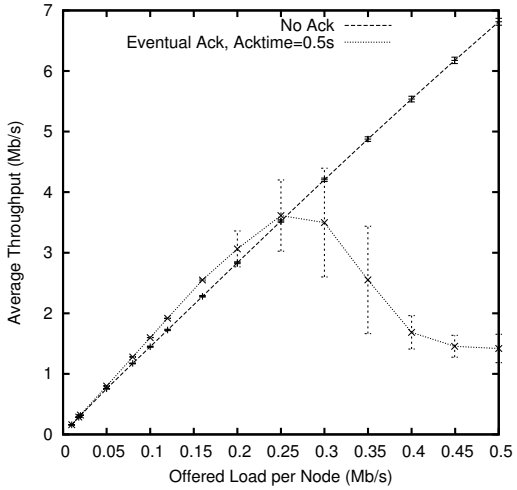
Figure 6.9: Eventual Ack performance on grid and random topologies. We see here that compared to the grid topology, Eventual Ack on random topologies performs poorly.



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Random
Ack Policy	Varied
Acktime	0.5 s
I	10
B	10
M	∞
K	∞

(d) Parameters

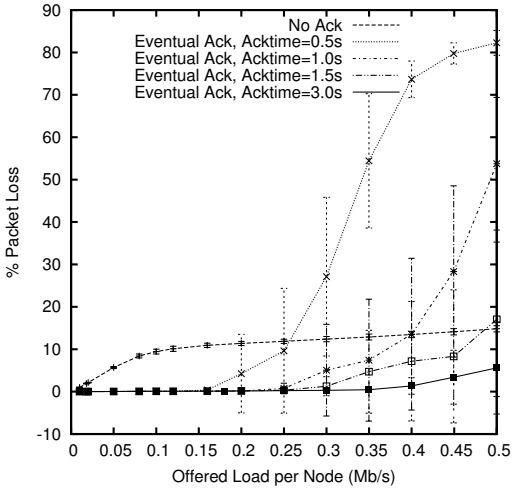
Figure 6.10: Eventual Ack policy compared to No Ack on random topologies. We see that Eventual Ack does not perform as well as No Ack under higher loads, as packets are retransmitted prematurely.

sections we vary each of them to determine their effects on system performance, in an attempt to improve performance on our random topologies. We will begin in Section 6.2.3 by investigating the eventual ack retransmission timeout value, *Acktime*. We will then adjust the inter packet spacing for simultaneous transmission, I, discussed in Section 4.2. After that, we will investigate in Sections 6.2.5 and 6.2.6 the MAC initial backoff period, B, and data queue size M, from Section 4.4. Finally, we will put all of this together in Section 6.2.7, where we will vary the capability of the multiuser detector, K, to determine what kind of capability we might need to achieve good results.

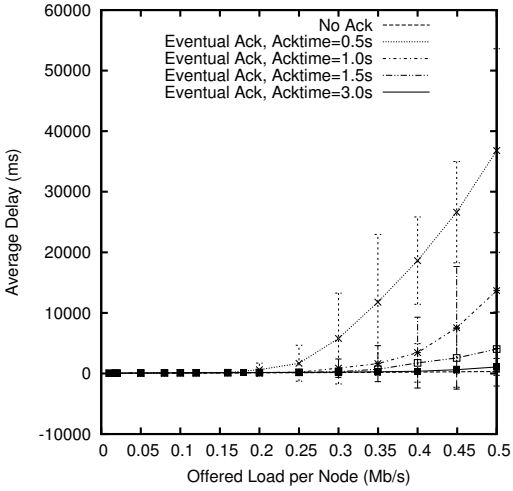
6.2.3 Effect of Varying Acktime

We tested various values for the worst case timer mechanism in the range $0.5s \leq Acktime \leq 3.0s$. These results are shown in Figure 6.11, and show that as the *Acktime* increased, system performance improved and eventually surpassed that of No Ack. This shows that as we remove the premature or unnecessary retransmissions from the channel, Eventual Ack was able to correctly identify only those packets that were actually lost, and retransmit only those.

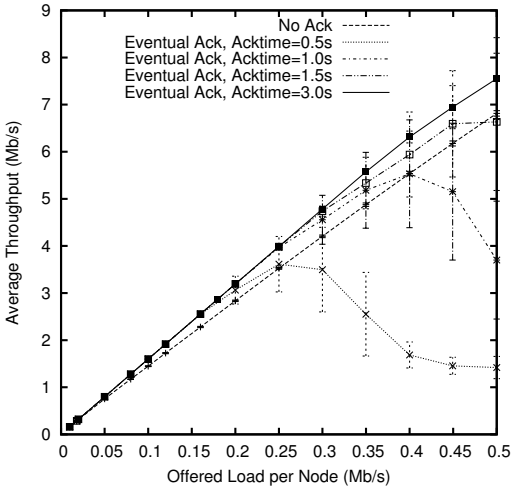
What is most interesting about these results is that as we increased the *Acktime*, all of our performance metrics were steadily improved. As *Acktime* increased, packet losses decreased, delay was reduced, and system throughput increased. Furthermore, the confidence intervals in our results became more narrow, indicating that not only was the system performing better, but was also more stable. With these observations in mind, we suggest that it may improve system performance even further if we disabled the retry timer altogether, and effectively set $Acktime = \infty$. If we recall, the *Acktime* retry timer mechanism was only intended as a guard against the edge case scenario in which all packets to a given neighbour node were lost. Rather than performing this function, it seems that the timer retry mechanism is actually more likely to cause unnecessary retransmissions, which in turn causes packet loss rather than alleviates it. The consequence of disabling this retry timer is that we will no longer be able to recover from the case when all packets to a given neighbour are lost, which means that it will have to be dealt with by upper layers



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Random
Ack Policy	Eventual Ack
Acktime	Varied
I	10
B	10
M	∞
K	∞

(d) Parameters

Figure 6.11: Evantual Ack with varying *Acktime*. We see here that as we increase the *Acktime*, the performance of the Eventual Ack policy begins to surpass that of the No Ack policy.

in the network stack. Given that the intended outcome of not protecting against this case in the MAC layer is that overall packet loss is reduced, we consider this consequence to be acceptable.

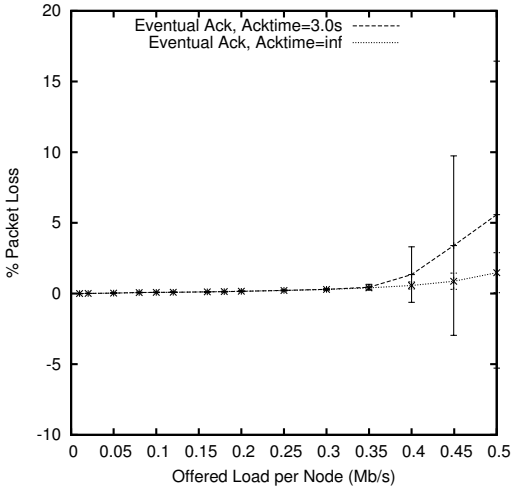
We see in Figure 6.12 the result of setting $Acktime = \infty$. In this figure, we see that for most of the tested range, the longer *Acktime* value did not significantly alter the results, but as the offered load per node approached 0.5 Mbps, we see that disabling the timer retry mechanism yielded superior packet loss and end to end delay results. Additionally, the confidence intervals for our results were smaller, indicating improved system stability.

From these results, we conclude that having the timer retry mechanism is unnecessary and even counterproductive. It is clearly not required in order to achieve high performance in our system, and the Eventual Ack policy appears to be able to accurately identify lost packets through the ordering of sent and acknowledged packets alone.

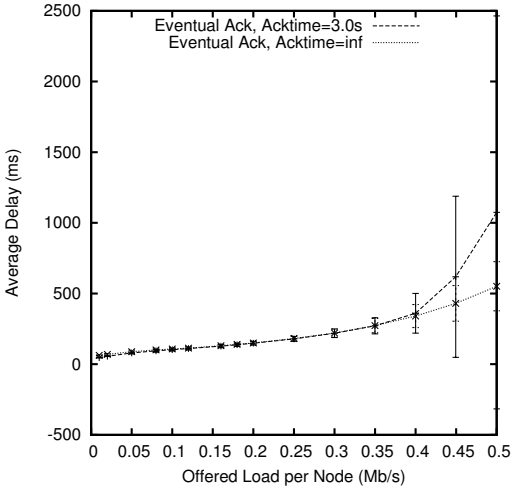
6.2.4 Effect of Varying Simultaneous Packet Spacing, I

We now turn our attention to the simultaneous packet spacing, I. Recall from Section 4.2 that I is the maximum number of header intervals between simultaneous packets transmitted from a single sender. Higher values of I will mean that simultaneous packets will be more spread out in time, and smaller values for I mean simultaneous packets will be more compressed. We have thus far presented all of our results with $I = 10$, which matches our value for the initial backoff period $B = 10$. This number was chosen arbitrarily, and we now investigate whether $I = 10$ is the optimal value, or if some other value yields improved performance.

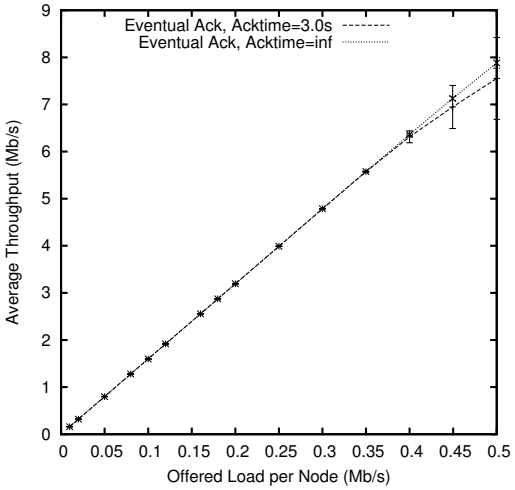
Our primary concern here is that if we reduce the value of I too much, then we may lose more packets to header collisions in the event that more than one node begins transmitting at about the same time within range of a common neighbour. Said differently, while we are less concerned about the hidden node problem in RP-CDMA , it is still possible for packet headers to collide, and in the hidden node situation packing simultaneous packet headers together more tightly may result in more header collisions. Alternately, if we choose a value of I that is too large,



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Random
Ack Policy	Eventual Ack
Acktime	Varied
I	10
B	10
M	∞
K	∞

(d) Parameters

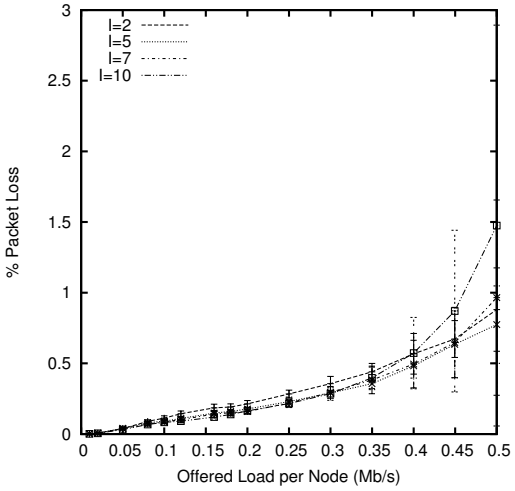
Figure 6.12: Eventual Ack with $Acktime = \infty$. We see here that disabling the timer retry mechanism in Eventual Ack yields superior metrics for packet loss, end to end delay and system throughput under high loads.

then simultaneous packets may be spaced somewhat farther apart, in which case the average number of simultaneous packets will fall and we will make less efficient use of the available time in which a given node may transmit.

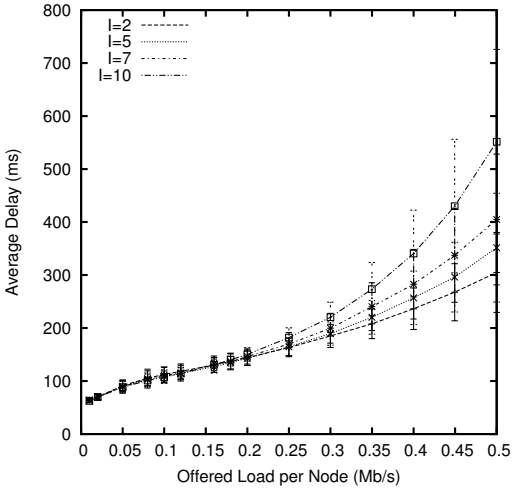
We tested various values for I in the range $2 \leq I \leq 10$, and show our results in Figure 6.13. We see in this figure that for progressively smaller values of I that performance improved. In particular, end to end delay was smaller for smaller I, while the effects on packet loss or throughput were less obvious. In order to see if there was any noticeable effect on packet loss for varying I, we expanded our offered load per node range out from 0.5 Mbps to 1 Mbps, which is equal to the channel speed.

We see in Figure 6.14 our results after expanding the range of our experiments. It is evident from these results that we can achieve better performance for smaller values of I, and so we choose $I = 2$ as our preferred value. Note that for $I = 2$, our algorithm for randomly selecting a number of backoff slots is constrained to always select 1, since we choose an integer slot in the range $[1, I)$, and so always choose 1 for $I = 2$. It is worth mentioning that we considered choosing $I = 0$, and thus not staggering simultaneous packet headers at all, but chose not to in order to provide some spacing between packet headers in which to allow the theoretical transceiver time to hand off one packet to the multiuser detector and prepare to sync to the next one, similar to the role of the SIFS time in 802.11 [48].

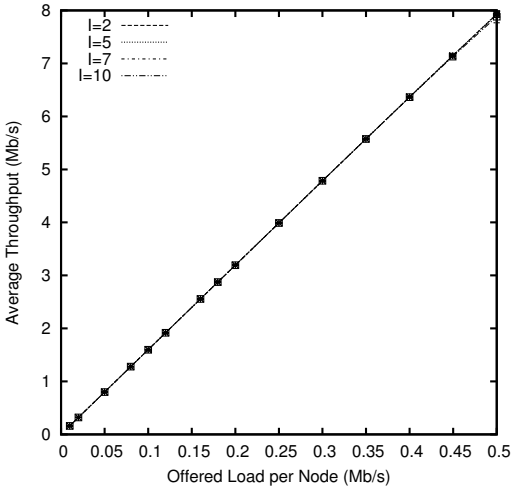
Of final note with these results, we can see in Figure 6.14c that the system maintained almost linear throughput to offered load per node with the inter-packet spacing set to $I = 2$. This is notable because for an offered load per node of 1 Mbps, which is the top of our tested range, each node in the network was able to access a capacity almost equal to the channel speed, which is also 1 Mbps. This demonstrates the potential of simultaneous transmission, as it allows transmitting nodes to fit large amounts of data into whatever slice of the shared channel they are able to access.



(a) Average Packet Loss



(b) Average Delay

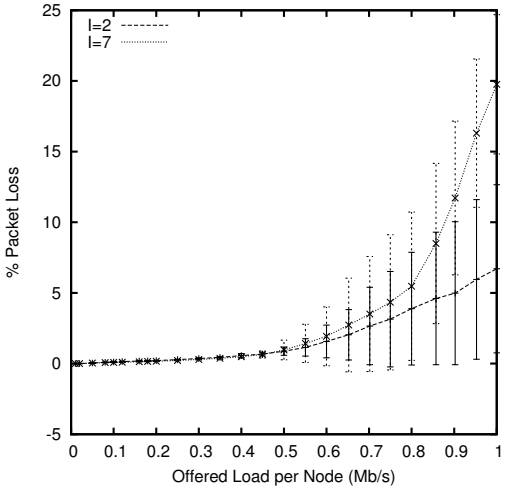


(c) Average System Throughput

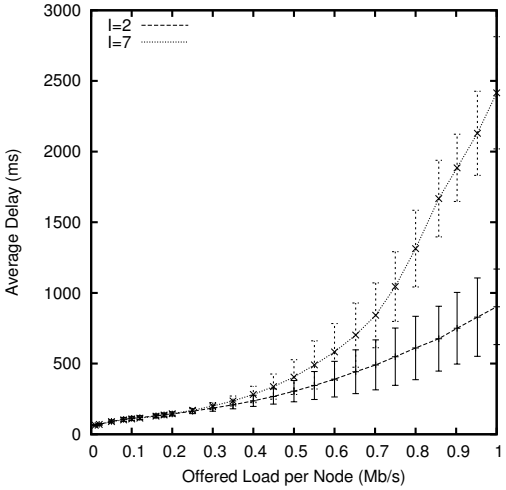
Topology	Random
Ack Policy	Eventual Ack
Acktime	∞
I	Varied
B	10
M	∞
K	∞

(d) Parameters

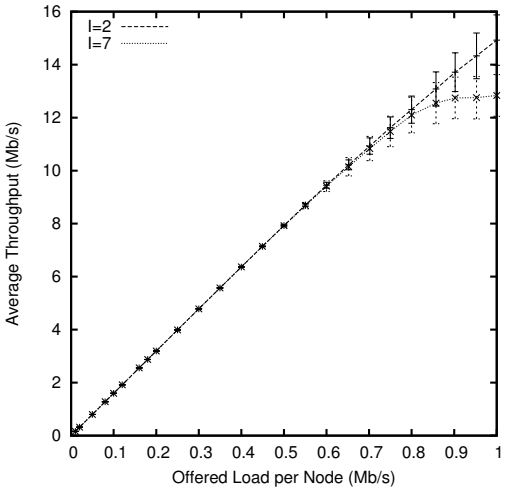
Figure 6.13: Eventual Ack with varying simultaneous packet spacing, I. We see here that shorter values for I provide improving system performance. For $I = 2$, simultaneous packets are uniformly spaced with one packet header between them.



(a) Average Packet Loss



(b) Average Delay

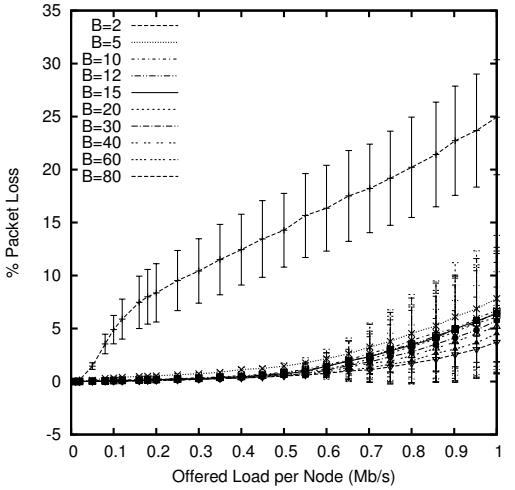


(c) Average System Throughput

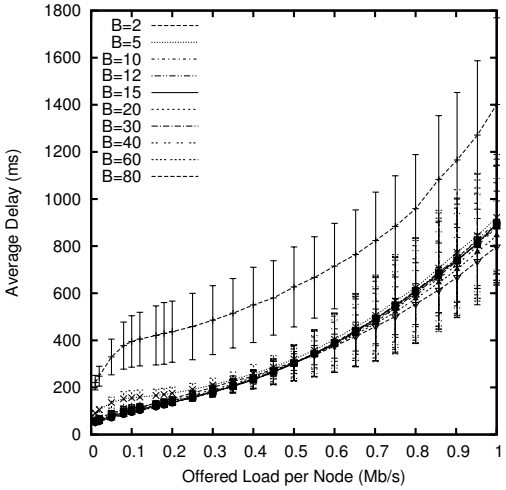
Topology	Random
Ack Policy	Eventual Ack
Acktime	∞
I	Varied
B	10
M	∞
K	∞

(d) Parameters

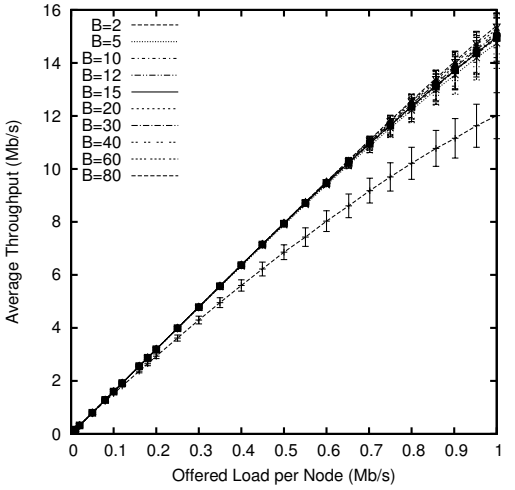
Figure 6.14: Eventual Ack with $I = 2$ and $I = 7$. We see here that for very high offered loads, that we achieve better performance with $I = 2$, rather than larger values.



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Random
Ack Policy	Eventual Ack
Acktime	∞
I	2
B	Varied
M	∞
K	∞

(d) Parameters

Figure 6.15: Eventual Ack with varying initial backoff, B.

6.2.5 Effect of Varying Initial Backoff Period, B

Having selected a preferred inter-packet spacing, I, we turn our attention to the initial backoff period, B. With a shorter initial backoff interval range we expect that the probability of more than one node starting to transmit at the same time will be higher.[3] We expect that this higher probability of multiple nodes transmitting at the same time will result in more lost packets due to both packet header collisions and packets arriving while the receiving node is in the transmit state.

[3]This expectation is derived in the Appendix Section A.2.3.

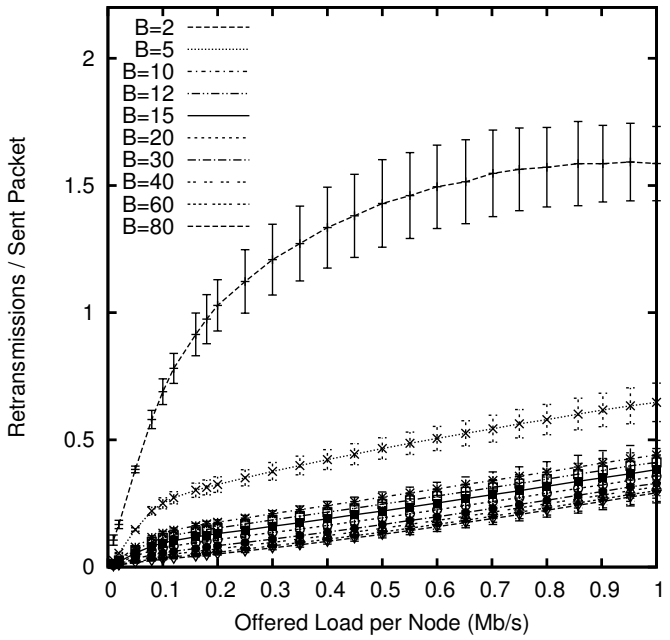


Figure 6.16: Number of retransmissions per sent packet for various values of B. We see here that for values of $B > 10$, the number of retransmissions is only marginally improved.

Our results for various values of B are shown in Figure 6.15. From these results, we can see that a value for $B = 2$ was clearly worse in every metric than our other tested values. For all other values we tested, our results for packet loss and delay were within error of each other, indicating that our system is less sensitive to the range of possible initial backoff periods. This is understandable if we consider that the job of the acknowledgement policy is to retransmit packets after they are detected to be lost. Thus, even if smaller values of B yielded increased numbers of lost packets, as long as the acknowledgement and retransmission mechanism was effective then those packets would eventually reach their intended receiver node and be counted.

It is instructive, therefore, to examine the number of retransmissions for each of our possible values for B. These results are shown in Figure 6.16, which shows the number of retransmissions per sent packet as a function of the offered load per node. We measured the number of retransmissions at the MAC level, and the number of sent packets at the application level, and so this metric tells us the average number of times each packet sent into the network was retransmitted. We see from these

results that for values of $B > 10$, the improvement in the number of retransmissions was marginal, while the overall system performance was approximately the same. If we substantially increased the initial backoff period B into the range of 60 or 80 header intervals we did note a decrease in the average number of packets lost under high loads and the average number of retransmissions per packet, but the effect on system throughput and end to end delay was less clear.[4]

We conclude from this discussion that our system can achieve reasonable performance with initial backoff periods in the range $B \geq 10$. For more dense node configurations than tested here we expect that larger initial backoff periods would benefit system performance, but that for our purposes setting $B = 10$ is sufficient to achieve high performance.

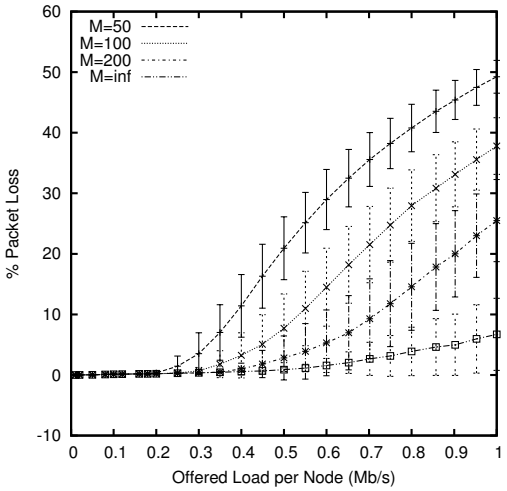
6.2.6 Effect of Limited Device Data Queue Size, M

Up to this point, we have considered systems with infinite data queues but recognize that this is an unrealistic assumption. We therefore wish to identify how limiting the size of the MAC data queue affects performance. We expect that as the MAC queue decreases in size that more packets will be dropped both as they arrive from the sending node application layer and from neighbouring nodes.

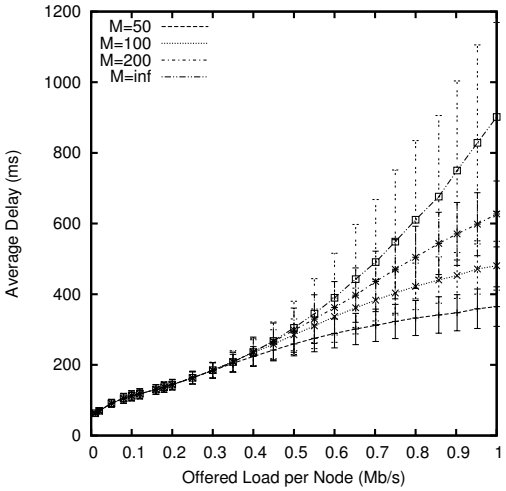
Our results are shown in Figure 6.17, which shows that as we increased the device queue length system performance increased towards the infinite queue length result, which was as expected. As we increased the length of the data queue the number of lost packets decreased and system throughput correspondingly increased, at the cost of higher end to end delay. This result makes sense when we consider that a shorter data queue means that fewer packets would be accepted into the system for transmission, but each accepted packet would have less time to wait as the overall number of packets accepted into the system (and hence the load) was smaller.

At this point in our performance study we have identified several parameters we can vary in order to increase the performance of our system, and now begin turning our attention towards what would achievable in a realistic device. To this end, we

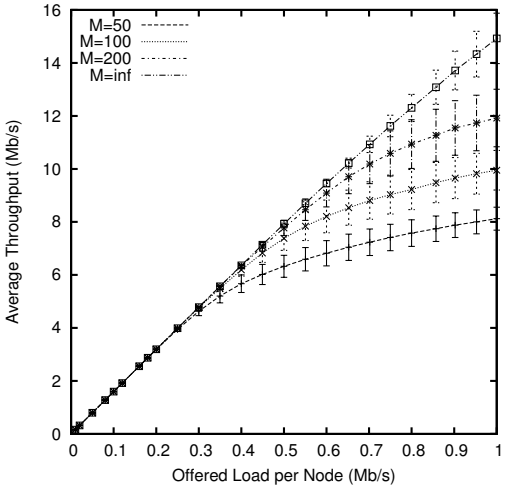
[4] In the Appendix, Figure A.7 shows that this result is attributable to the probability of more than one node starting transmission at the same time becoming insensitive to the initial backoff period for values in the range $B > 30$.



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Random
Ack Policy	Eventual Ack
Acktime	∞
I	2
B	10
M	Varied
K	∞

(d) Parameters

Figure 6.17: Eventual Ack with limited MAC data queue size, M.

wish to select a value for the maximum queue length, M, that we can adopt going forward in our experiments and which is not objectionable from a memory resource point of view. If we consider the default Queue object in ns-3, we see that it has a default maximum length of 100 [2], which for our 1500 B sized packets, works out to 150 kB of queue memory, plus some extra for packet headers, *etc.*. This seems reasonable, and so we adopt $M = 100$ for our subsequent experiments.

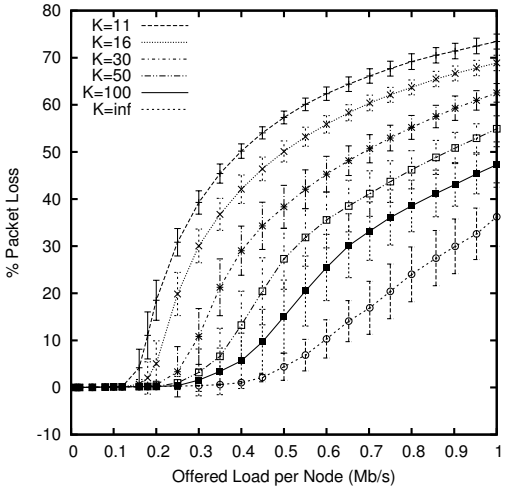
6.2.7 Effect of Limited Multiuser Detectors

Finally, we turn our attention to limiting values for the multiuser detector, K. We began investigating the limitations imposed by realistic systems in the previous section, and found that limiting the amount of buffering available in the sending nodes has the expected effect on system performance, and that larger queues result in better packet delivery rates at the cost of higher latency. In this section, we investigate limitations on the multiuser detector capability, K, and try to determine what kind of detector we would need in order to approach the ideal case where $K = \infty$.

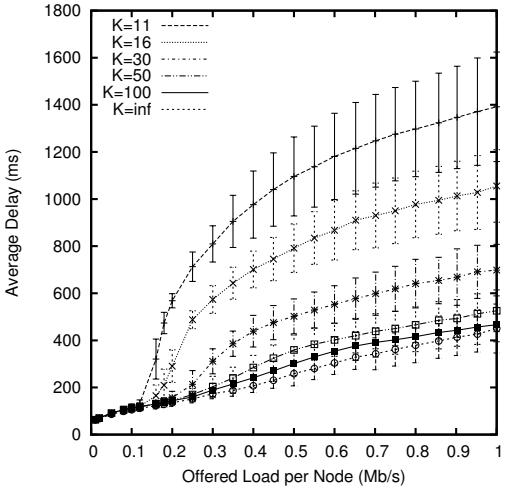
Figure 6.18 shows how the multiuser detector capability, K, impacted the system performance. In this figure, we show our ideal results against results with limited multiuser detector capability. Typical values for K found in the literature are in the range $5 \leq K \leq 20$ [43], though values of $100 \leq K \leq 500$ are sometimes discussed [17, 36]. Our results show that as we increased the detector capability system performance steadily approached the ideal, $K = \infty$, result. Specifically, for each value of K tested, system performance matched the ideal result for some range of offered loads per node, and this range increased in size as we increased the detector capability. These results are as expected, though when we compare them to the results shown in Figure 6.4, where we varied K on the grid topology, we can see the net effects of the random topology and particularly the limited data queue size.

6.2.8 Comparision to 802.11 CSMA

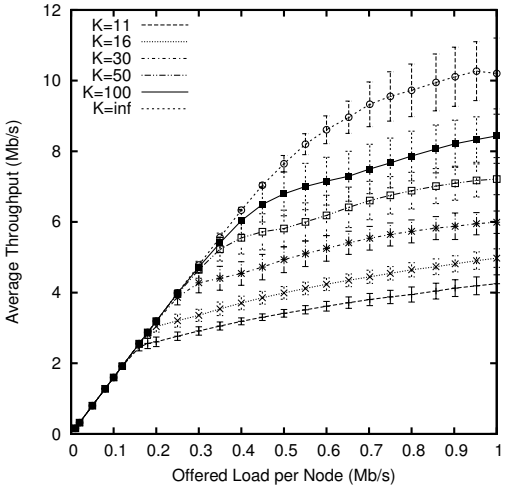
Finally, we can compare our RP-CDMA device performance with that of 802.11 on our random topologies. Again, we limit ourselves to $K = 11$ in order to make a fair



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Random
Ack Policy	Eventual Ack
Acktime	∞
I	2
B	10
M	100
K	Varied

(d) Parameters

Figure 6.18: Eventual Ack with limited multiuser detector and data queue size $M = 100$.

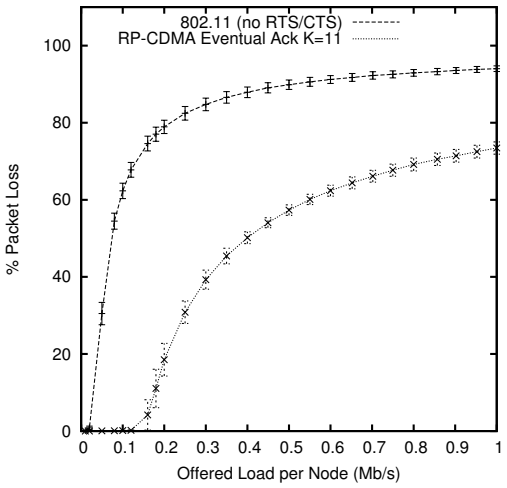
bandwidth comparison, and also limit our device to a data queue size of $M = 100$, with parameters $B = 10$, $I = 2$. These results are shown in Figure 6.19. We see in these results that the RP-CDMA device again performed better than 802.11 across the tested range.

6.3 Summary

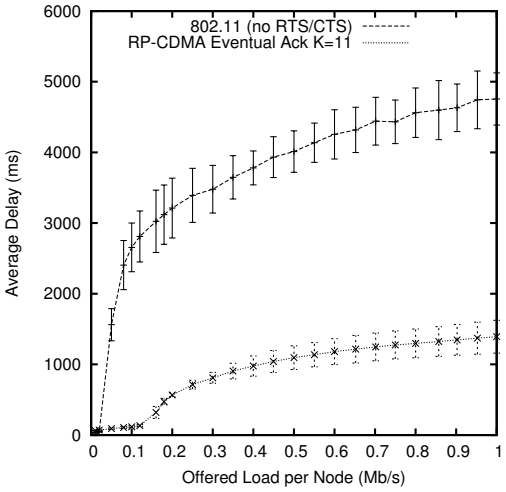
In this chapter we started with an initial assessment of RP-CDMA on a simple grid topology, and found that RP-CDMA system performance exceeded that of 802.11 even without any acknowledgement policy. We experimented with two acknowledgement policies, one which guaranteed packet acknowledgement within a fixed time frame and one that that did not. We found that the fixed time acknowledgement policy did not perform well, primarily because of the need to cut off multiple packet receptions in order to meet acknowledgement time guarantees. In contrast, the Eventual Ack policy, which did not guarantee when acknowledgements would be sent, performed very well on our grid topology and effectively reduced packet losses to negligible values even under high loads while maintaining low end to end delay.

On random topologies, we investigated the effect of varying the *Acktime* for our Eventual Ack acknowledgement mechanism and found that timed retransmissions were unnecessary, and could actually be counterproductive. We adjusted the amount of spacing between simultaneously transmitted packets, I, and found that it was better to try to fit more packets into less time than to worry about packet header collisions between different transmitting nodes. When we varied the initial backoff period before transmission was initiated, B, we found that overall system performance was not highly sensitive to values for $B > 10$ on our 16 node topologies. After investigating these system parameters, we found that the RP-CDMA system could maintain an almost linear throughput response to offered load per node up to the channel speed of the network.

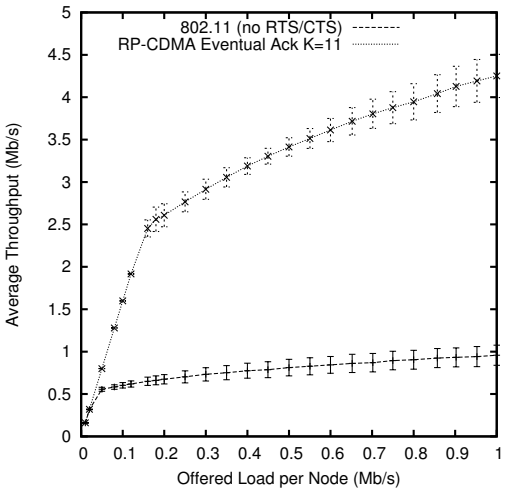
After exploring the limits of system performance with ideal receivers, we applied limitations to both the data queue size in the MAC, M, and the capability



(a) Average Packet Loss



(b) Average Delay



(c) Average System Throughput

Topology	Random
Ack Policy	Eventual Ack
Acktime	∞
I	2
B	10
M	100
K	11

(d) Parameters

Figure 6.19: Eventual Ack compared to 802.11 on random topologies. We see here that the RP-CDMA device performs better than the 802.11 device across the tested range.

of the multiuser detector, K. We found that limiting the resources available to the system reduced its performance under high loads, but that as we allocated more resources to the system in terms of queue space or multiuser detector capability that system performance approached the ideal result. Finally, we showed that even with limited capability our RP-CDMA device offered superior performance to 802.11 under identical conditions.

From our results, we conclude that RP-CDMA is a promising wireless multiple access protocol for multihop ad hoc networks. By sending packet payloads in private channels and employing a multiuser detector to enable the spreading of simultaneous packets over varying transmission power levels, RP-CDMA can provide wireless nodes with reliable links to their neighbours. We have investigated the effects of several device parameters on system performance on both grid and random topologies, and found that it is possible to set parameter values which simultaneously enabled both low packet loss and low latency. We have seen that with reasonable resource limits on the device data queue and multiuser detector capability the RP-CDMA device can continue to provide relatively high system throughput while maintaining low latency, and in particular can deliver superior performance to 802.11 in the same bandwidth.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this dissertation we have investigated the application of RP-CDMA to wireless multihop ad hoc networks with the aim of improving wireless link reliability and network performance. We have proposed extending the standard RP-CDMA protocol with simultaneous transmission and payload channel acknowledgements, with the goal of improving network throughput and reliability while maintaining a relatively light load on the common packet header channel. We have described a simple MAC that can be built on top of the RP-CDMA multiple access method that maintains low latency and high throughput even under high load, and presented a simple acknowledgement policy that reliably delivers packets while maintaining these desirable qualities.

We have performed simulations of our RP-CDMA network device using ns-3 on a 16 node network in both grid and random topologies. Our results showed that the RP-CDMA net device is capable of maintaining high packet delivery and throughput in addition to low latency even under heavy load - up to the point where each node in the network offered a load equal to the channel speed. We have compared our results to the popular 802.11 CSMA/CA multiple access method, and found our RP-CDMA device offers superior performance and reliability across our tested range.

In our introduction we suggested that the steadily increasing performance and reliability of wired networks is rooted in the nature of the wired link, and asked

how a wireless link might have the same qualities. We identified RP-CDMA as a promising wireless link option because of the novel way in which it transmits packets in randomly assigned private channels without any coordination required between sender and receiver. By eliminating the need to coordinate between nodes for medium access, we were able to propose a simple MAC protocol and acknowledgement policy that enabled high system throughput and low delay while reliably delivering packets over multiple network hops. Our simulation results suggest that a reliable wireless link can enable reliable, high performance multihop ad hoc networks.

7.2 Future Work

In this dissertation we have universally presented results employing an ideal receiver which neglected interference. While CDMA is resistant to interference, it would be informative to consider its effects in our simulations. As future work we would like to verify our device performance with interference effects included. At the same time as we consider interference, it would also be useful to apply error correction and information coding mechanisms in order to more closely approximate a realistic multiuser detector.

If we step up from the network device level, we believe that it would be worthwhile to investigate end to end reliability mechanisms between the sending and receiving nodes at the network transport level. We briefly investigated TCP performance in Section 6.1.4, and found that while RP-CDMA improved performance compared to 802.11, it did not approach the performance obtained with UDP. We believe that TCP is worthy of further investigation, particularly an investigation into the effect of our Eventual Ack policy on performance, and investigating modifications to TCP which may improve performance such as TCP Vegas [7]. While modifications to TCP have already been shown to to marginally improve performance over wireless links [14], we do not believe we should limit ourselves to TCP-only solutions and believe it would be worthwhile to investigate alternate end to end reliability mechanisms that specifically leverage the properties of the RP-CDMA

link.

In addition to the ad hoc network context, it would be informative to investigate the performance of RP-CDMA in an infrastructure network setting with a base station. In situations where network traffic is primarily in one direction, a highly capable base station - perhaps with a radio for both the downlink and uplink - may offer high performance. In particular, it would be worthwhile to investigate the performance of our RP-CDMA device when applied to the conference problem in which many clients attempt to access a base station simultaneously. Some work has already been done that is applicable to this context, such as in [16, 17], and it would be worthwhile to undertake a performance study with simulated network devices.

In this work, we examined only simple acknowledgement policies which acknowledged every packet. We justified this by noting that payload channel acknowledgements and simultaneous transmission made the impact of acknowledgements on the system acceptable for our purposes. With this said, more sophisticated acknowledgement policies, such as collective or cumulative acknowledgements, are worth investigating, particularly in the context of low capability multiuser detectors.

Finally, in the course of designing our MAC protocol and doing our performance study, we received many suggestions for possible modifications or improvements which might improve performance. These suggestions included modifications to the Phy header designed to reduce the size or number of headers in the common channel, modifications to the MAC transmission algorithm which could selectively ignore some incoming transmissions if they were deemed not important, and modifications to the sending of acknowledgement packets which would see them squeezed out during packet reception without having to drop packets. We resisted the urge to experiment with these modifications at the time in the name of maintaining simplicity, but we acknowledge that it would be worthwhile to investigate these suggestions in the future.

Bibliography

[1] 3GPP - Specifications. `http://www.3gpp.org`.

[2] ns-3 Network Simulator. `http://www.nsnam.org/docs/doxygen/index.html`.

[3] N Abramson. THE ALOHA SYSTEM. In *Proceedings of the Fall Joint Computer Conference, November 17-19*, 1970.

[4] N Abramson. Multiple access in wireless digital networks. *Proceedings of the IEEE*, 82(9):1360–1370, 1994.

[5] A Acharya, A Misra, and S Bansal. MACA-P: a MAC for concurrent transmissions in multi-hop wireless networks. *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, pages 505–508, 2003.

[6] A Al Hanbali, E Altman, and P Nain. A survey of TCP over ad hoc networks. *IEEE Communications Surveys & Tutorials*, 7(3):22–36, 2005.

[7] L.S Brakmo and L.L Peterson. TCP Vegas: end to end congestion avoidance on a global Internet. *Selected Areas in Communications, IEEE Journal on*, 13(8):1465–1480, 1995.

[8] MV Burnashev, CB Schlegel, WA Krzymien, and Z Shi. Analysis of the Dynamics of Iterative Interference Cancellation in Iterative Decoding. *Problems of Information Transmission*, 40(4):297–317, 2004.

[9] Shan Chen, B Bensaou, and Ka Lok Hung. Performance of Different TCP Variants in IEEE 802.11 WLAN and the TCP-WOW Algorithm. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–6, 2009.

[10] Jung Choi, Mayank Jain, Kannan Srinivasan, Phil Levis, and Sachin Katti. Achieving single channel, full duplex wireless communication. *MobiCom 2010*, pages 1–12, 2010.

[11] T Clausen and P Jacquet. RFC 3626 - Optimized Link State Routing Protocol (OLSR). *IETF RFC3626*, 2003.

[12] Federal Communications Commission. Electronic Code of Federal Regulations: Title 47: Telecommunication.

[13] J Domčnech, A Pont, and J Sahuquillo. A user-focused evaluation of web prefetching algorithms. *Computer Communications*, 30:2213–2224, 2007.

[14] S.M ElRakabawy, C Lindemann, and M.K Vernon. Improving TCP performance for multihop wireless networks. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, pages 684–693, 2005.

[15] S Floyd and T Henderson. RFC 2582. *IETF RFC2582*, April 1999.

[16] M Ghanbarinejad, C Schlegel, and P Gburzynski. Adaptive Probabilistic Medium Access in MPR-Capable Ad-Hoc Wireless Networks. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–5, 2009.

[17] M Ghanbarinejad and Christian Schlegel. Controlled random access with multipacket reception and traffic uncertainty. *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1238–1242, 2010.

[18] Holtzman. DS/CDMA successive interference cancellation. *Spread Spectrum Techniques and Applications, 1994. IEEE ISSSTA '94., IEEE Third International Symposium on*, pages 69–78 vol.1, 1994.

[19] Seung Min Hur, Shiwen Mao, Y Hou, Kwanghee Nam, and J Reed. A Location-Assisted MAC Protocol for Multi-Hop Wireless Networks. *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pages 322–327, 2007.

[20] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, New York, 1991.

[21] Kyu-Tae Jin and Dong-Ho Cho. A MAC algorithm for energy-limited ad-hoc networks. *Vehicular Technology Conference, 2000. IEEE VTS-Fall VTC 2000. 52nd*, 1:219–222 vol.1, 2000.

[22] Kyu-Tae Jin and Dong-Ho Cho. Multi-code MAC for multi-hop wireless ad hoc networks. *Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th*, 2:1100–1104 vol.2, 2002.

[23] R Jurdak and C Lopes. A survey, classification and comparative analysis of medium access control protocols for ad hoc networks. *Communications Surveys & Tutorials*, 6(1):2–16, 2004.

[24] R Kempter and C Schlegel. Capacity and QoS analysis for a novel packet based wireless access system. *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, 3:1432–1436 Vol. 3, 2003.

[25] Roland Kempter. *MODELING AND EVALUATION OF THROUGHPUT, STABILITY AND COVERAGE OF RP-CDMA IN WIRELESS NETWORKS*. PhD thesis, University of Utah, November 2006.

[26] Roland Kempter, Peiman Amini, and Behrouz Farhang-Boroujeny. Enhancing the Performance of Random Access Networks with Random Packet CDMA and Joint Detection. *EURASIP Journal on Advances in Signal Processing*, 2009:1–17, 2009.

[27] Leonard Kleinrock. *Queueing Systems*. Volume 1: Theory. John Wiley & Sons, New York, 1975.

[28] P Kota and C Schlegel. A wireless packet multiple access method exploiting joint detection. In *Communications, 2003. ICC '03. IEEE International Conference on*, pages 2985–2989, 2003.

[29] S Kumar and V Raghavan. Medium Access Control protocols for ad hoc wireless networks: A survey. *Ad Hoc Networks*, 2006.

[30] Mathieu Lacage and Thomas Henderson. Yet another network simulator. *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, page 12, 2006.

[31] Pierre L'Ecuyer, Richard Simard, E. Jack Chen, and W. David Kelton. An Object-Oriented Random-Number Package with Many Long Streams and Substreams. *Operations Research*, 50(6):pp. 1073–1075, 2002.

[32] Jhong S. Lee and Leonard E. Miller. The effect of path diversity (RAKE) on link reliability in CDMA cellular systems-a realistic assessment of system capacity. *Personal, Indoor and Mobile Radio Communications, 1996. PIMRC'96., Seventh IEEE International Symposium on*, 3:1130–1134 vol.3, 1996.

[33] Joo Ghee Lim, Chun Tung Chou, and Nyandoro. A Cut-through MAC for Multiple Interface, Multiple Channel Wireless Mesh Networks. *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pages 2373–2378, 2007.

[34] K Mittal and E Belding. RTSS/CTSS: mitigation of exposed terminals in static 802.11-based mesh networks. *Wireless Mesh Networks, 2006. WiMesh 2006. 2nd IEEE Workshop on*, pages 3–12, 2006.

[35] S Nagaraj and C Schlegel. A channel accessing scheme with joint detection receivers in ad hoc networks. In *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, pages 381–386, 2004.

[36] S Nagaraj, D Truhachev, and C Schlegel. Analysis of a Random Channel Access Scheme with Multi-Packet Reception. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5, 2008.

[37] Kitae Nahm, Ahmed Helmy, and C C Jay Kuo. TCP over multihop 802.11 networks: issues and performance enhancement. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. ACM Request Permissions, May 2005.

[38] Dejan M Novakovic and Miroslav L Dukic. Evolution of the power control techniques for DS-CDMA toward 3G wireless communication systems. *Communications Surveys & Tutorials, IEEE*, 3(4):2–15, 2000.

[39] T Paul and T Ogunfunmi. Evolution, insights and challenges of the PHY layer for the emerging ieee 802.11n amendment. *Communications Surveys & Tutorials, IEEE*, 11(4):131–150, 2009.

[40] S Ray, J B Carruthers, and D Starobinski. RTS/CTS-Induced Congestion in Ad Hoc Wireless LANs. In *Wireless Communications and Networking, 2003. WCNC-03*, pages 1516–1521. IEEE, 2003.

[41] C Schlegel. CDMA with Partitioned Spreading. *Communications Letters, IEEE*, 11(12):913–915, 2007.

[42] C Schlegel, R Kempter, and P Kota. A novel random wireless packet multiple access method using CDMA. *Wireless Communications, IEEE Transactions on*, 5(6):1362–1370, 2006.

[43] Christian Schlegel and Eric Bouton. Multi-code wireless packet random access. *MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010*, pages 731–736, 2010.

[44] Christian Schlegel and Alex Grant. Coordinated Multiuser Communications. Springer, 2006.

[45] S Sesay, Z Yang, and J He. A survey on mobile ad hoc wireless network. *Information Technology Journal*, 3(2):168–175, 2004.

[46] Jungmin So and Nitin Vaidya. Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver. *ACM international symposium on Mobile Ad Hoc Networking and Computing*, pages 222–233, 2004.

[47] LAN MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Information technology- Telecommunications and information exchange between systems- Local and metropolitan area networks- Specific requirements–Part 15.4: Wireless MAC and PHY Specifications for Low-Rate WPANs. IEEE, September 2006.

[48] LAN MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks— Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standard, 2007.

[49] LAN MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Information Technology—Telecommunications and information exchange between systems—LANs and MANs—Specific requirements—Part 11: WLAN MAC and PHY Specifications—Amendment 5: Enhancements for Higher Throughput. *IEEE*, pages 1–536, October 2009.

[50] LAN MAN Standards Committee of the IEEE Computer Society. IEEE Std 802.3-2008, Section One. *IEEE*, pages 1–671, January 2009.

[51] William Stallings. *Data and Computer Communications*. Prentice Hall, Upper Saddle River, NJ, 07458, ninth edition, 2011.

[52] Hang Su and Xi Zhang. A Spreading Code MAC Protocol for Multi-Hop Wireless Ad Hoc Networks. *Communications, 2007. ICC '07. IEEE International Conference on*, pages 3590–3595, 2007.

[53] Z Tang and J Garcia-Luna-Aceves. A protocol for topology-dependent transmission scheduling in wireless networks. *Wireless Communications and Networking . . .*, pages 1333–1337 vol.3, 1999.

[54] M Veyseh, J.J Garcia-Luna-Aceves, and H.R Sadjadpour. Cross-Layer Channel Allocation Protocol for OFDMA Ad Hoc Networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6, 2010.

[55] Kaixin Xu, M Gerla, and Sang Bae. How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks. *IEEE Global Telecommunications Conference*, 1:72–76 vol.1, 2002.

[56] Shugong Xu and Tarek Saadawi. Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks? *Communications Magazine*, June 2001.

[57] Hujun Yin and Siavash Alamouti. OFDMA: A Broadband Wireless Access Technology. *Sarnoff Symposium, 2006 IEEE*, pages 1–4, 2006.

[58] Chenxi Zhu and MS Corson. A five-phase reservation protocol (FPRP) for mobile ad hoc networks. *Wireless Networks*, 7(4):371–384, 2001.

Appendix A

Theoretical Model

In this appendix we perform analysis on our system to better understand how it behaves. We begin with a discussion of what our MAC looks like from the queueing model perspective in Section A.1, starting with a general description in Sections A.1.1 and A.1.2, followed by a simplified model that we can validate in the simulator in Section A.1.3. Following our simplified analysis, we estimate the effect on system performance of a scaled up system consisting of 16 nodes in a grid configuration in Section A.2. We begin with a discussion of the effective load on the network given multihop in Section A.2.1, followed by a discussion of the expected transmission time for some number of packets given simultaneous transmission in Section A.2.2. Finally, we finish in Section A.2.3 with a discussion of the probability of any node beginning to transmit after the channel becomes unoccupied, and in particular the probability that more than one node begins transmitting at the same time.

A.1 Queueing Analysis

In general, an accurate queueing model for our MAC is more complex than we can analyse in closed form. We can, however, describe the system in general and then use simplifying assumptions to reduce our general model to a simplified one which submits to closed form analysis. In this section, we will first describe how our system looks from the queueing model point of view, and then proceed with a simplified model which we can verify in the simulator.

A.1.1 General Description

As we saw in Section 4.4, our MAC utilizes a type of carrier sense mechanism coupled with a uniform random backoff (Algorithm 4.3) in order to determine when to send a packet down to the Phy for transmission. For each packet, the MAC waits until the Phy determines that it is safe to begin transmitting, then waits a randomly chosen backoff period, and if the Phy still indicates it is safe to transmit then the packet is sent to the Phy, where it is transmitted into the channel.

We can represent this as a queueing model, which we show in Figure A.1. In this figure, packets arrive at the MAC from higher layers with rate λ. Upon arrival at the MAC, each packet enters a first come first served queue, shown on the left of Figure A.1. Once a packet reaches the front of the queue, it enters service at the first of three servers. The first server, G, has a General distribution. This server is responsible for determining when it is safe to transmit, which in practice means waiting until there are no packets being received at the device Phy. Thus, the service time in the server G ranges from zero, corresponding to the case where no packets are being received at the Phy, up to the time required in order for one or more neighbouring nodes to finish transmitting some number of packets, which we will call T_{rx}. In the case of infinitely capable multiuser detectors, the time required for a neighbouring node to finish transmitting is the time required for that node to clear its own data queue, which is determined by the queue length and the channel speed. After completing service in the server G (so, once the Phy determines it is safe to transmit), the packet moves into another server, U, whose service time follows a Uniform Discrete distribution. This is the random backoff period. This server has a service time distributed uniformly in the discrete range $[1, B)T_h$,[1] where $T_h = L_h/S$, which is just the number of bits in a packet header, L_h, divided by the channel speed, S. When service is complete in the server U, the packet proceeds with probability p_s to a third server, D, with Deterministic service time. This is the Phy, where the time required to service the packet is simply the time required to

[1] We ignore for the moment the effects of simultaneous transmission, which would have the discrete range be either $[1, B)T_h$ or $[1, I)T_h$ depending on whether or not the node was transmitting more than one packet.

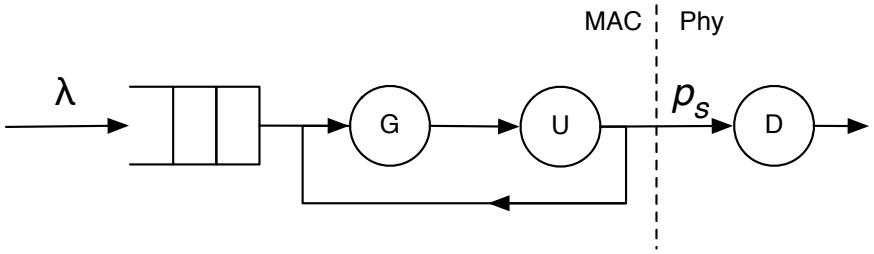


Figure A.1: A general queue model of our MAC. Packets enter the system at the left, and exit the system into the channel on the right.

transmit the packet into the channel. So for the server D, the service time is just $T_{hp} = L_{hp}/S$. The probability p_s is the probability that the Phy will indicate that it is safe to transmit at the end of service at server U, so p_s is the probability that the Phy does not begin receiving a packet from a neighbouring node in that time period. Thus, the probability p_s is a function of the number of neighbours for a given node, the fraction of those neighbours who have a packet to transmit, and the probability that any one of those neighbours with a packet begins transmitting before service at U is complete. In the event that the packet completes service at server U but does not proceed to server D, the packet will return to the first server in the chain, G. Once a packet has finished service at server D then the next packet from the data queue enters service at server G.

A.1.2 Simultaneous Transmission

If we add simultaneous transmission into our model, then we need to change the service time functions for each of our servers to accommodate the case when a node transmits more than one packet simultaneously. When transmitting simultaneously, the service time at G is zero, at U is uniformly in the range $[1, I)T_h$, and at D becomes just the time required to transmit the packet header, T_h. Additionally, we must add K deterministic servers to the end of our model in order to represent the transmission time of packet payloads up to the capability of the multiuser detector. We thus arrive at the following system:

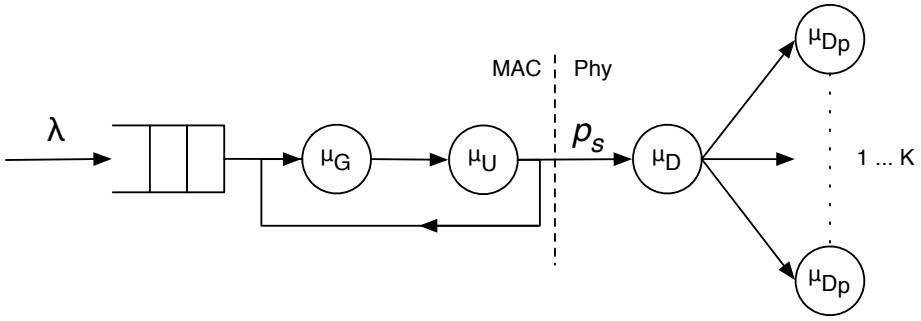


Figure A.2: A general queue model of our MAC, incorporating simultaneous transmission. Packets enter the system at the left, and exit the system into the channel on the right. Note the K deterministic queues at the end of the system, which are responsible for transmitting up to K packet payloads.

$$\mu_G = \begin{cases} 0 & : \text{When IDLE} \\ 0 & : \text{When TX\_PAYLOAD} \\ T_{rx} & : \text{Otherwise} \end{cases} \quad \text{(A.1)}$$

$$\mu_U = \begin{cases} [1, I)\, T_h & : \text{When TX\_PAYLOAD} \\ [1, B)\, T_h & : \text{Otherwise} \end{cases} \quad \text{(A.2)}$$

$$\mu_D = T_h \quad \text{(A.3)}$$

$$\mu_{D_p} = T_p \quad \text{(A.4)}$$

This system is shown in Figure A.2. The system can thus service up to K packets concurrently in the payload servers indicated by $D_{p[1...K]}$, and as a packet moves from the header transmission server, D, to one of the payload servers, the next packet from the queue enters service at server G.

A.1.3 Simplified Model

The system represented in Figure A.2 is difficult to represent in closed form, particularly because the service time for any one packet depends partially on the probability that some other node is transmitting (μ_G) and on p_s, which represents that probability that the packet proceeds from server U to server D instead of returning the server G. We can, however, create a simplified model in which these complications are removed. If we create a system with a single sending node and single receiving node, we can create a system in which the service time $\mu_G = 0$ for all packets in the sending node, $p_s = 1$, and, due to simultaneous transmission, the

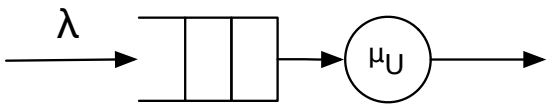


Figure A.3: A simplified queueing model for the single sending node configuration.

service times in the payload servers D_p are irrelevant. What remains is the service times μ_U and μ_D. Because μ_D is fixed, we can simply adjust the uniform range in U upward by one packet header time T_h. Finally, if we set $I = B$, then what remains is the simplified system shown in Figure A.3 and given by:

$$\mu_U \quad = \quad [2, B+1)\, T_h \tag{A.5}$$

With Poisson arrival rate, λ, this system then becomes a M/G/1 queueing system with G being a uniform discrete random distribution, which we can analyse in closed form. We know from Kleinrock [27] that the average number of customers in the M/G/1 system is given by

$$\bar{q} = \rho + \rho^2 \frac{(1 + C^2)}{2(1 - \rho)} \tag{A.6}$$

Where $\rho = \lambda \bar{x}$ as usual, with λ being the Poisson arrival rate and $\bar{x}$ being the expected service time. C^2 is the squared coefficient of variation for service time, $C^2 = \sigma^2/(\bar{x})^2$, which is completely specified by the first and seconds moments of the service time distribution. For the discrete uniform distribution in the range $[2, B+1)$, we can calculate the these values directly:

$$\bar{x} \quad = \quad \sum_{i=2}^{B} i T_h p_i \tag{A.7}$$

$$\sigma^2 \quad = \quad \sum_{i=2}^{B} \left(i T_h - \bar{x}\right)^2 p_i \tag{A.8}$$

Where $p_i = 1/((B+1) - 2)\ \forall\ i$, which is just the uniform probability of choosing any of the values in the range $[2, B+1)$.

If we choose $B = 10$, $\lambda = 1/0.0012$ Hz, $T_h = L_h/S = 144$ b$/1 \times 10^6$ bps $= 0.000144$ s then we can use formulas A.6 to A.8 to calculate specific values:

$$
\begin{aligned}
\bar{x} &= 0.000864s \\
\sigma^2 &= 1.3824 \times 10^{-7} \\
\rho &= 0.72 \\
C^2 &= 0.1851 \\
\bar{q} &= 1.81714
\end{aligned}
$$

Where $\bar{q}$ is the expected number of packets in the system, including packets in the queue and in service. The expected number of packets in only the queue is given by [27]:

$$\bar{q}_q = \bar{q} - \rho = 1.0971 \tag{A.9}$$

We can simulate this simplified system and compare the simulated results with our predictions above. In our simulated implementation, we measure the number of packets waiting in the MAC queue, which does not include any packets being transmitted by the Phy. Specifically, a packet leaves the MAC queue and is passed to the Phy when it passes from the server U to the server D, which we have indicated in Figures A.1 and A.2 with a dashed line. Thus, when we measure the number of packets in the MAC queue during an experiment, we expect that some of the time the packet currently 'in service' is in the MAC queue, and some of the time it is in the Phy. Thus, we expect that some of the time the number of packets measured in the MAC queue will represent all of the packets in the system ($\bar{q}$) and some of the time it will represent only the number of packets in the queue ($\bar{q}_q$). Since a packet spends T_h time in the Phy (server D), then we expect that the probability of finding a packet in service at the Phy to be $T_h/\bar{x}$, and so we can combine equations A.6 and A.9 to get the expected number of packets measured in the MAC queue during an experiment:

$$q = \frac{T_h}{\bar{x}}\bar{q}_q + \left(1 - \frac{T_h}{\bar{x}}\right)\bar{q} \tag{A.10}$$

Which, for the same parameters we list above, gives us an expected MAC queue length in our simulation of $q = 1.6971$.

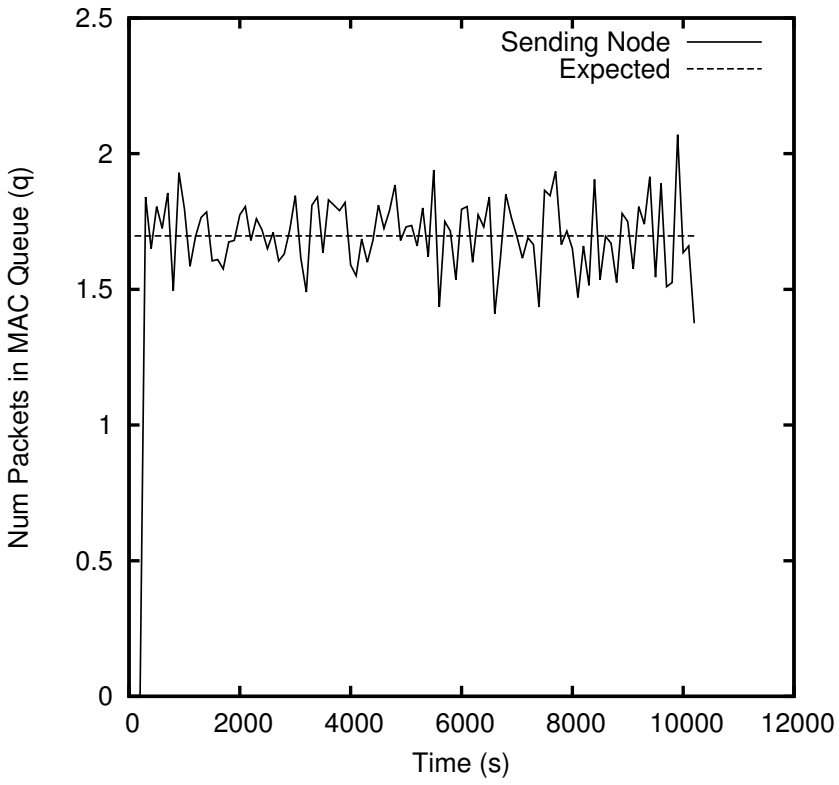


Figure A.4: The measured MAC queue length for the single sending node example.

We performed 10 independent trials of this single sender experiment. In each trial we measured the MAC queue length of the sending node every 5 seconds over 10000 seconds, and then took the average of the 10 trials. The result is shown in Figure A.4, where we have plotted the average of each 20 measurements (so the average queue length over 100 seconds) for the sending node along with the expected value. In this figure we see that the average queue length of the sending node fluctuates around the expected value, and when we calculate the average and 95% confidence interval for our measured values, we find $q = 1.7023 \pm 0.02727$, which is within error of our expected value of $q = 1.6971$.

We performed the same experiment with a different packet arrival rate, $\lambda = 1/0.024$, which yields expectations:

$$\begin{aligned} \bar{x} &= 0.000864s \\ \sigma^2 &= 1.3824 \times 10^{-7} \end{aligned}$$

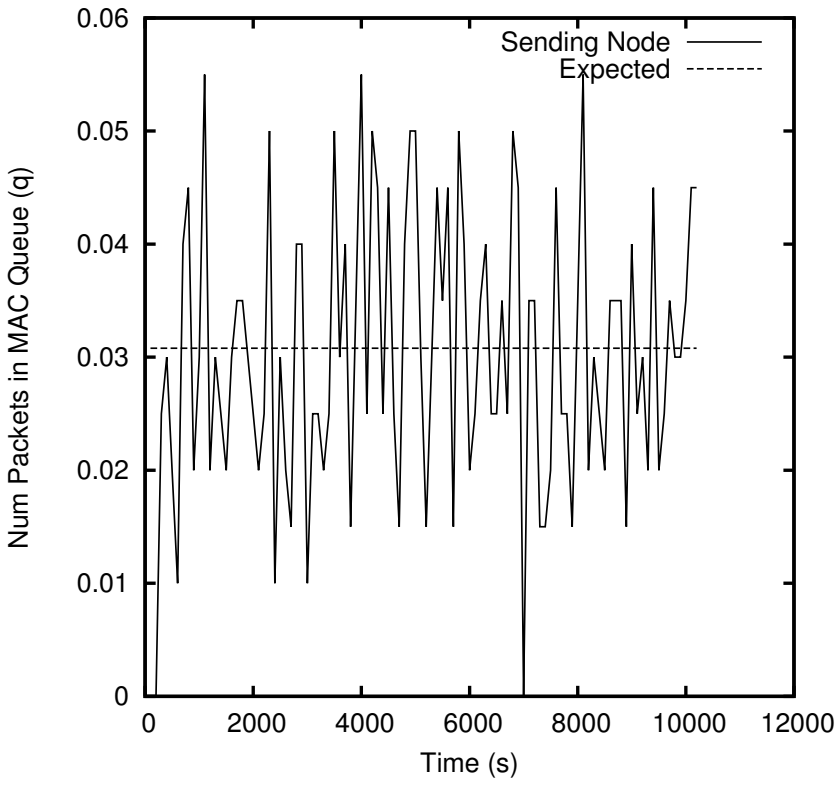


Figure A.5: The measured MAC queue length for the single sending node example with light load.

$$\begin{aligned} \rho &= 0.036 \\ C^2 &= 0.1851 \\ \bar{q} &= 0.03679 \\ q &= 0.03079 \end{aligned}$$

Our results from this experiment are shown in Figure A.5. We can see again in this result that the measured queue length fluctuates around the expected value, and when we take the average of our measurements and the 95% confidence intervals, we get $q = 0.03065 \pm 0.002406$, which is again within error of our expected value of $q = 0.03079$.

From these results, we conclude that our simplified queueing model accurately represents our simulation in this example.

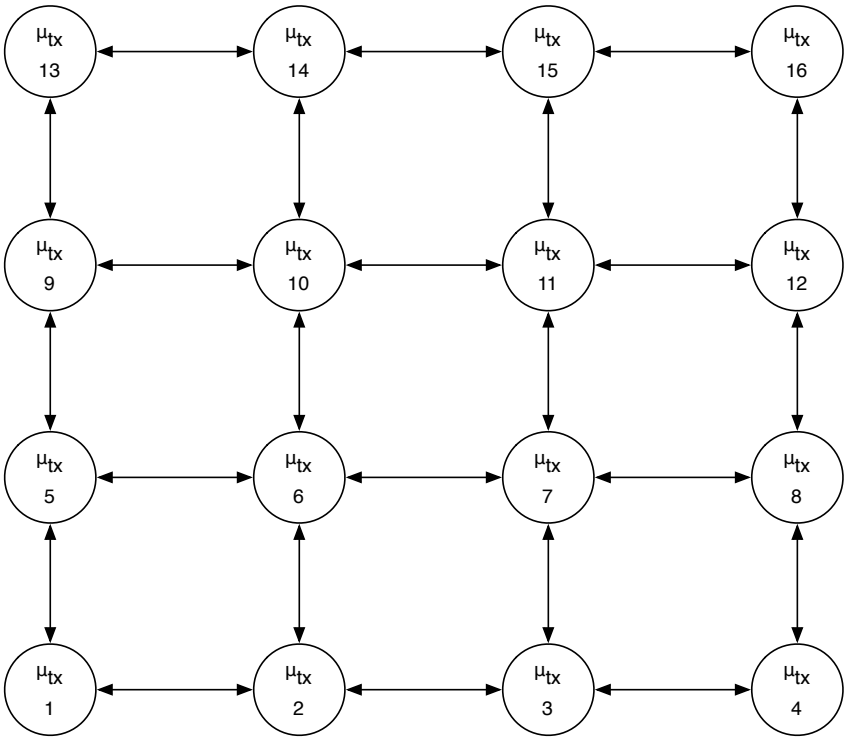


Figure A.6: Grid Topology Queueing System. Here we represent the each node's transmission queue (Figure A.2) as a single server with service time μ_{tx}, along with the node number.

A.2 Scaling Up

Having verified our simulation on a simplified example, we can now discuss the larger system. Our aim in this section is to investigate the expected values for the general queueing model parameters, λ, T_{rx} and p_s from Section A.1.2. Because we cannot come to a closed form solution of the general MAC queueing model, we will consider a simplified system of nodes, and discuss how to estimate these parameters.

If we consider 16 nodes placed in a grid configuration, which can communicate only with their neighbours on the same X and Y axes, then we get the queueing system shown in Figure A.6. In this figure, we represent the node MAC queues using a single server with service time μ_{tx}, and each node is numbered from 1 to 16.

If each node experiences Poisson packet arrivals with rate λ, and each packet is addressed to a uniformly randomly chosen node in the network, then we can es-

timate the expected number of hops per packet, and thus the expected load on the system as a function of the aggregate input load from each node. We perform this analysis in Section A.2.1. In Section A.2.2, we analyse our simultaneous transmission mechanism in order to derive the effective data rate in the presence of concurrently outgoing packets. With this estimate, and our estimate for effective load, we can discuss the expected service time, μ_{tx}. Finally, we can perform an analysis of the probability that any node begins transmission after an idle period in order to obtain an estimate of the probability p_s, which we do in Section A.2.3.

A.2.1 Expected Load with Multihop

Given our grid topology with $N = 16$ nodes, we can calculate the expected number of hops for any packet directly. For any node, n, the expected number of hops for a packet which is addressed to a uniformly random other node in the network is given by:

$$\overline{H_n} = \frac{1}{N-1} \sum_{i=1}^{N} h_{n \to i} \tag{A.11}$$

Where $h_{n \to i}$ is the shortest number of hops from node n to node i. For the grid topology in Figure A.6, we can count $h_{n \to i}$ directly:

n	Destination Node i															
1	0	1	2	3	1	2	3	4	2	3	4	5	3	4	5	6
2	1	0	1	2	2	1	2	3	3	2	3	4	4	3	4	5
3	2	1	0	1	3	2	1	2	4	3	2	3	5	4	3	4
4	3	2	1	0	4	3	2	1	5	4	3	2	6	5	4	3
5	1	2	3	4	0	1	2	3	1	2	3	4	2	3	4	5
6	2	1	2	3	1	0	1	2	2	1	2	3	3	2	3	4
7	3	2	1	2	2	1	0	1	3	2	1	2	4	3	2	3
8	4	3	2	1	3	2	1	0	4	3	2	1	5	4	3	2
9	2	3	4	5	1	2	3	4	0	1	2	3	1	2	3	4
10	3	2	3	4	2	1	2	3	1	0	1	2	2	1	2	3
11	4	3	2	3	3	2	1	2	2	1	0	1	3	2	1	2
12	5	4	3	2	4	3	2	1	3	2	1	0	4	3	2	1
13	3	4	5	6	2	3	4	5	1	2	3	4	0	1	2	3
14	4	3	4	5	3	2	3	4	2	1	2	3	1	0	1	2
15	5	4	3	4	4	3	2	3	3	2	1	2	2	1	0	1
16	6	5	4	3	5	4	3	2	4	3	2	1	3	2	1	0

We can calculate the expected number of hops for any packet in the system by taking the average of $\overline{H_n}$ over all n:

$$\overline{H} = \frac{1}{N} \sum_{n=1}^{N} \overline{H_n} \tag{A.12}$$

And for our grid topology, we get $\overline{H} \approx 2.6667$. This means that for each packet sent into the network, we expect that it will have approximately $\overline{H}$ hops to its destination, or that each packet sent into the network will have to be transmitted approximately $\overline{H}$ times.

In terms of the arrival rate, λ, if each node experiences Poisson packet arrivals at rate λ, then the system arrival rate is $\lambda_s = N\lambda$. If each packet experiences $\overline{H}$ hops, then the system arrival rate is expected to be increased to $\lambda_s = \overline{H}N\lambda$, not counting any retransmissions.

A.2.2 Expected Service Time with Simultaneous Transmission

In this section, we estimate the expected service time for a single node transmitting some number of fixed size packets into the network. This analysis will be approximate because we assume fixed size packets, but we can nonetheless explore the factors that contribute to the expected service time.

We begin with the transmit time for X simultaneous packets, and refer back to Figure 4.1, which shows three simultaneous packets being transmitted into the channel. Let L_h be the length of the packet header, L_p be the length of the packet payload, and $L_{hp} = L_h + L_p$. The start of each packet header is separated from the end of the previous one by a random uniform value in the range $i \in [1, I)L_h$. The expected value of this uniform random value is given by $E[i] = \frac{I}{2}L_h$. Thus, the expected number of bits between the start of one packet header and the start of the next is given by $L_I = L_h + E[i] = \left(1 + \frac{I}{2}\right) L_h$. In terms of time, the expected time between the start of one packet and the start of the next is therefore $T_I = L_I/S = \left(1 + \frac{I}{2}\right) \frac{L_h}{S}$, where S is the data rate of the transmitter.

Now, when we transmit X packets simultaneously, assuming for the moment that X is less than the multiuser detector limit, $X < K$, then we can calculate the expected time from the start of the first packet to the end of the last packet,

assuming that all packets have equal length given by L_{hp}. In this case, there are $X-1$ intervals of length T_I between the start of the first packet and the start of the last packet, plus the time to transmit the last packet, T_{hp}. Thus, we expect that the time required to transmit X packets with simultaneous transmission is given by:

$$T_X = (X-1)\left(1+\frac{I}{2}\right)\frac{L_h}{S} + \frac{L_{hp}}{S} \tag{A.13}$$

If we drop the assumption that $X < K$, we can extend this analysis. First, we consider the quantity $X_k = \frac{L_{hp}}{L_I}$, which is the length of a full packet divided by the inter-packet spacing. This quantity represents the maximum number of packets that we expect to see being transmitted simultaneously, since the $X_k + 1$ packet will begin transmission after the first packet is completed. Thus, for systems in which $X_k = \frac{L_{hp}}{L_I} < K$, we expect simultaneous transmission will not reach the limit of the multiuser detector, and Equation A.13 holds for all X.

In the event that $X_k > K$ and $X > K$, then our MAC will transmit up to K packets, then stop adding new packets until all K are completely transmitted, after which it will proceed through the normal MAC backoff procedure before initiating another transmission. In this case, the time to transmit X packets, T_X, is less straightforward, since it depends on the probability of a neighbouring node initiating a transmission in the backoff period, which is given by $(1-p_s)$ in our general queueing model.

In terms of our queueing model from Section A.1.2, T_X corresponds to the service time μ_G, since the expected time to transmit X packets, T_X, is the same as the time required to receive them. Given the expected number of packets waiting to be transmitted from a given sending node, $\bar{q}$, we can calculate $T_{\bar{q}}$, and a packet which enters service while the node is in the receive state can expect to wait $T_{rx} = T_{\bar{q}}/2$ for the reception to finish. We can combine this with the probability of a node being in the receive state at any given time, p_{rx}, to get the expected service time $E[\mu_G] = p_{rx}\frac{T_{\bar{q}}}{2}$.

In this section we have approximated the expected transmit time for a number of packets via simultaneous transmission, T_X. We have also identified the maximum number of simultaneous packets that can be in transmission at the same time,

X_k, and identified that for multiuser detectors with capability $K > X_k$, that simultaneous transmission will allow for a closed form expected transmission time for any number of packets, given by Equation A.13, under the assumption of fixed size packets. Finally, we have related the expected transmission time of some number of packets to the expected waiting time in neighbouring nodes, μ_G.

A.2.3 Probability of Transmitting After Idle Period

We now turn our attention to the probability of a node commencing transmission after an idle period, which is denoted by p_s in our general queueing model. Our approach here will be to calculate the probability of a given node in a group of fully connected peers which all have a packet to send choosing the lowest random uniform number in the range $[1, B)$ after the channel becomes idle. The node which chooses the shortest backoff period will commence transmission first, and the other peers will wait. In addition to the probability of beginning transmission, p_s, we will also be able to calculate the probability of more than one node choosing the same shortest random backoff interval and transmitting at the same time, which is undesirable in the ad hoc context as it may result in packet loss.

We begin by describing our simplified example. Suppose we have a node, Node 1, with N neighbours. Suppose Node 1 is transmitting and the N neighbours are all receiving. Each of the receiving N nodes has a packet to transmit. When Node 1 finishes transmitting, each of the N neighbours will uniformly randomly select an integer number of header slots to back off before transmitting, $b \in [1, B)$. If some number of nodes, n, all select the same value for b, and that value is the smallest b chosen among all N nodes, then those n nodes will begin transmitting and the end of the backoff period. If $n = 1$, then only one node will begin transmission, and if $n > 1$, then more than one node will begin transmission at the same time.

The probability p_s is given by the probability that a specific node chooses some backoff period, $P(b) = \frac{1}{B-1}$, and the probability that all of the other nodes choose the same backoff period or a larger one, $P(N-1, \geq b) = \left(\frac{(B-1)-(b+1)}{B-1}\right)^{N-1}$. Therefore:

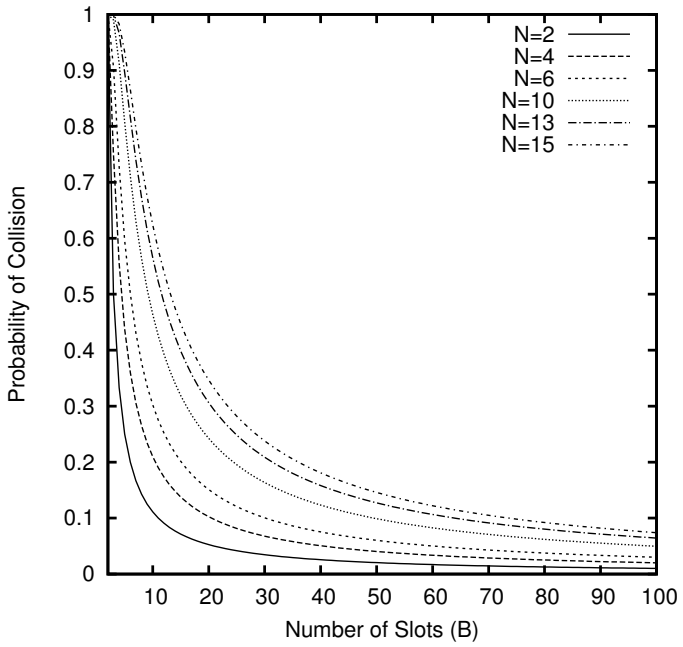


Figure A.7: The probability of more than one node transmitting in the same slot, $b \in [1, B)$, in a group of N nodes.

$$p_s = \left(\frac{1}{B-1}\right)\left(\frac{(B-1)-(b+1)}{B-1}\right)^{N-1} \tag{A.14}$$

Given the number of nodes in a group and the probability that a node has a packet to transmit at a given time, we can use Equation A.14 to calculate the probability that any one of them will begin transmitting after the channel goes idle.

From this point, it is straightforward to calculate the probability of more than one node starting transmission after an idle period. The probability of some n nodes choosing the same value for b is given by $P(n,b) = \frac{1}{B-1}^n$, and there are $\binom{N}{n}$ ways to select the n nodes. The probability that the selected b is the smallest b selected of all N nodes is given by $P(N-n, > b) = \left(\frac{(B-1)-b}{B-1}\right)^{N-n}$. We can therefore calculate the probability that some number of nodes all select the same, smallest, b by summing these probabilities over all n and b. Therefore:

$$P(>1,b) = \sum_{b=1}^{B-1}\sum_{n=2}^{N}\binom{N}{n}\left(\frac{1}{B-1}\right)^n\left(\frac{(B-1)-b}{B-1}\right)^{N-n} \tag{A.15}$$

Finally, the probability that only one node begins transmitting after an idle period is given by one minus the probability of more than one transmitting $P(1,b) =$

$(1 - P(> 1, b))$.

Equation A.15 shows the probability of more than one node beginning to transmit in the given slot $b \in [1, B)$. When this happens, we can expect that their first packets will collide in their headers and be lost. Additionally, any packets which are intended for any of the transmitting nodes will be lost, as the intended receiver will be in the transmit state when the packet arrives. The probability of packets being lost is an important quantity in our system, as we want to minimize the probability that more than one node starts transmitting in a peer group at the same time. We can visualize how Equation A.15 varies with the number of nodes and the size of the backoff window by calculating it for selected values of N and B, which we show in Figure A.7. In this figure, we see that the probability of more than one node starting transmission at the same time drops off rapidly as the number of slots increases, and is generally smaller for smaller peer groups, as we would expect. In particular, for smaller values of N, increasing the number of backoff slots beyond $B > 30$ yields only marginal reductions in the probability of more than one node starting to transmit at the same time. Note that in our Grid configuration, we have $2 \leq N \leq 4$, and in any topology with only 16 nodes will never have $N > 15$.

In this section, we have calculated the probability that a node will begin transmitting after an idle period, p_s, and calculated the probability that more than one node begins transmission in a given peer group. The probability p_s is one of the elements in our queueing model, and given a probability that each node in a group has a packet to transmit at any time, we can calculate the probability that any one of them begins transmission after the channel becomes unoccupied. The probability that more than one node begins transmission at the same time is an important metric for our simulations, as we are interested in keeping this value as small as possible to prevent packet losses.

A.3 Summary

In this appendix we have explored a queueing model of our MAC, both on its own and in a grid configuration system. We began with a specification of a queueing

system representing our MAC, including the various servers involved as a packet proceeded from the queue, into the random backoff period, and then finally into the channel. This model had several parameters, including the arrival rate, λ, the time a node can expect to wait before an incoming transmission was completed, T_{rx}, and the probability that a node begins transmitting after the channel becomes unoccupied, p_s. While we did not find a closed form solution for the performance characteristics of our general queueing system, we were able to obtain a closed form solution for a simplified model, which we verified in our simulator. After verifying the simplified model, we discussed how we expect the general system parameters to behave. To this end, we identified the expected load multiplier for a grid system in the presence of multihop, and therefore identified how we expect the actual arrival rate, λ_s, to be related to the input rate. We then identified the expected transmission time for a quantity of packets in the presence of simultaneous transmission, and related it to the expected waiting time, T_{rx}. Finally, we investigated the probability that a node begins transmitting after the channel becomes unoccupied, p_s, and identified how to calculate this probability given a number of nodes with packets to transmit. Finally, we investigated the probability that more than one node begins transmitting at the same time, which is related to the expected amount of packet loss in our system.