RATE SELECTION FOR MIMO LINKS

BY

STEPHEN DAWIT ABRAHAM

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Adviser:

Professor Nitin H. Vaidya

ABSTRACT

Over the years, multiple-input multiple-output (MIMO) technology has become more prevalent in modern wireless communication systems. MIMO systems take advantage of multiple antennas at the transmitters and receivers, as well as multipath propagation, to provide more reliable, higher capacity wireless links. As a result, these links can support a set of much higher data rates than single-input single-output (SISO) links.

However, it is not always optimal for a node to transmit at its largest supported rate. Depending on various factors, it may be more efficient for a node to transmit at one of its lower transmission rates. One metric that can be used to determine the optimal transmission rate for a packet over a MIMO link is goodput.

In this thesis, we derive a model for a MIMO link between two wireless nodes. This model includes a method for simulating successful packet transmission based on the distribution of the link's capacity. In addition, the model uses the goodput metric to select the optimal transmission rate for a packet from a MIMO link's set of achievable rates. Through the use of MATLAB and modifications to the Network Simulator (NS-2), our MIMO link model is incorporated into NS-2 and simulations are executed for experimental data.

To my parents and brothers

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Nitin Vaidya for his guidance throughout the entire research process. I would also like to thank Craig Wilson, Jonathan Ligo, and Ghazale Hosseinabadi for taking the time to provide me with valuable input. Their knowledge and insight has been greatly appreciated during this process. To the new friends I have made, thanks for making this such a memorable experience.

Finally, I would especially like to thank my mother, father, brothers, and friends for their love and support over the past two years. They have always been there for me and I am forever grateful.

This work is funded in part by the National Science Foundation award number CNS 08-31670.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACK	Acknowledgment
CDF	Cumulative Density Function
DCF	Distributed Coordination Function
DIFS	Distributed Interframe Space
MAC	Medium Access Control
MIMO	Multiple-Input Multiple-Output
NAV	Network Allocation Vector
NS-2	Network Simulator
PDF	Probability Density Function
PLCP	Physical Layer Convergence Procedure
SIFS	Short Interframe Space
SISO	Single-Input Single-Output
SNR	Signal-to-Noise-Ratio
SVD	Singular Value Decomposition
TCL	Tool Command Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WLAN	Wireless Local Area Network

CHAPTER 1

INTRODUCTION

Over the years, multiple-input multiple-output (MIMO) technology has become more prevalent in modern wireless communication systems. MIMO systems take advantage of multiple antennas at the transmitters and receivers, as well as multipath propagation, to provide more reliable, higher capacity wireless links. This technology has already been incorporated into a number of wireless protocols such as IEEE 802.11n, WiMAX, and 4G LTE.

Most wireless protocols support a set of predefined, discrete transmission rates. For instance, under the IEEE 802.11n protocol, nodes operating with two streams can transmit at the following rates: 14.4, 28.9, 43.3, 57.8, 86.7, 115.6, 130, and 144.4 Mbps.[1] However, before transmitting a packet across a link, a decision needs to be made on which of the available rates to transmit with. In order to make such a decision, some defined metric needs to be used for comparison purposes.

Two metrics that can be used for comparison among a set of rates are throughput and goodput. Throughput is the measurement of bits transmitted over a period of time, while goodput is the measurement of data bits transmitted over a period of time. The goodput is smaller than the throughput since some of the transmitted bits are overhead. Goodput allows for a more realistic comparison in the sense that it only considers the payload portion of a packet.

[1] This is for a 20 MHz channel using a 400 ns guard interval.

In this thesis, we briefly discuss MIMO, IEEE 802.11, rate adaptation, and NS-2 in Chapter 2. In Chapter 3, we derive a model for a MIMO link between two wireless nodes, which allows us to generate both a probability and a cumulative density function (PDF and CDF) for a given link based on a set of link parameters. These distributions are then used to determine the packet reception rate for any given set of transmission rates. Using this information, we calculate the transmission time of a packet at each available rate and the associated goodput. The rate with the highest associated goodput, or the "goodput optimal" rate, is then selected as the rate to use for transmission. In Chapter 4 we discuss modifications made to NS-2 that allow for our MIMO model and rate selection algorithm to be incorporated into simulations. The results of our simulations are analyzed in Chapter 5, and in Chapter 6 we discuss how the work presented in this thesis can be expanded upon.

CHAPTER 2

BACKGROUND

MIMO technology uses multiple transmit and receive antennas to achieve higher reliability and larger capacities for wireless links. MIMO systems make use of two techniques in particular to achieve this increase in performance: spatial multiplexing, and diversity.

2.1 Multiple-Input Multiple-Output

Spatial multiplexing occurs when a transmission across a link is split into multiple independent streams of data. Each data stream is transmitted simultaneously over different transmit antennas [1]. The receiver, however, is responsible for demultiplexing and decoding each stream according to its unique spatial signatures. Spatial multiplexing is limited by the number of antennas at both the transmitter and receiver. In particular, this impacts the number of possible streams that can be used during transmission. For example, given a link with M transmit antennas and N receive antennas, which is referred to as an M x N link, the maximum number of streams possible is min(M, N). By using spatial multiplexing, the result is a linear increase in the link's capacity relative to min(M, N), which allows for larger data rates. No additional power or bandwidth is required in order to achieve this increase in capacity [2].

The other method of transmission, diversity, increases the reliability of the

wireless link by sending multiple copies of the same signal from transmitter to receiver. The idea behind this technique is that even though each replica could experience deep fading, the probability such a scenario occurs is far less than if only a single copy was transmitted [3]. This ensures that the destination receives the transmitted signal with higher reliability.

There are three primary types of diversity schemes that are used in MIMO systems: time diversity, frequency diversity, and spatial diversity. For time diversity, a signal is transmitted repeatedly in different time slots. Frequency diversity occurs when the same signal is transmitted over different channels with frequencies that are adequately spaced apart from one another [4]. Finally, spatial diversity refers to the use of multiple antennas at either the transmitter or receiver, with proper spacing in between each of the antennas. If there are more antennas at the receiver than at the transmitter, this is known as receive diversity. If the opposite is true, then it is referred to as transmit diversity; in this mode, each transmit antenna sends the same data over the channel. For each type of diversity described, the goal is to have many signal paths that are independent of one another such that each path experiences different fading.

These transmission strategies are the fundamental concepts MIMO technology uses to achieve more reliable, higher capacity links. Research [1], [3], [4] has been conducted to compare the effectiveness of the different modes and determine which transmission scheme is optimal under various conditions.

2.2 IEEE 802.11

Wireless local area networks (WLANs) follow a set of specifications and standards defined by IEEE 802.11, which was initially released in the late 1990s.

Since then, there have been many modifications and additions to the standards which have resulted in several distinct IEEE 802.11 protocols. Most recent versions of IEEE 802.11 support higher data rates, with MIMO being first incorporated in IEEE 802.11n.

To prevent and handle collisions between nodes in an asynchronous network, there needs to be a defined protocol to determine which node transmits at what time. As a result, medium access control (MAC) is handled in IEEE 802.11 through the use of the distributed coordination function (DCF) [5]. Before a node can begin transmitting data, it must first wait for a period of time known as the Distributed Coordination Interframe Space (DIFS). After the MAC layer is notified that a transmission needs to be performed, the node must detect that the medium is idle for the DIFS period before contending for the medium in the contention window. The contention window is simply a range of integer values. If the transmitting node detects the medium is free, waits the DIFS period, and then detects the medium is still free, the node transmits. However, if the medium is detected as busy, after waiting the entire DIFS period the transmitting node randomly selects one of the integer values in the contention window and uses that number as the amount of time slots it will wait until attempting to retransmit. When the payload has been received by the receiver, an amount of time known as the Short Interframe Space (SIFS) must pass [6]. This allows the receiver time to process the data and respond with an acknowledgment (ACK). The amount of transmission time the ACK requires will depend on the rate it is sent at.

In regard to the payload, the Physical Layer Convergence Procedure (PLCP) prepares the frame for transmission by appending a PLCP preamble and PLCP header to the beginning of the frame. The preamble is a set of symbols used by the receiving node for synchronization and the header includes

a field that informs the receiver of the rate the payload will be sent at. The PLCP bytes take a fixed amount of time to be transmitted.

The amount of time it takes for a packet to be transmitted under the IEEE 802.11 protocol will be affected by the DCF and PLCP as mentioned above. This is important as it will be factored into our goodput calculations in order to determine the optimal transmission rate.

2.3 Rate Adaptation and Selection

When delivering a packet from one node to another, it is possible to select the transmission rate according to the characteristics of the link that is being used [7]. This is known as rate adaptation. Many rate adaptation schemes (e.g., [8], [9], [10], [11], [12], [13]) have been developed and studied, often with the goal of maximizing network throughput. Typically, in IEEE 802.11 systems the channel state information at the transmitter is limited, which is important to consider because this is where selection of transmission rate occurs [10]. In our rate selection scheme we make an assumption that is considered ideal. We assume that the channel state of a link between two non-mobile nodes is static. In particular, the signal-to-noise ratio (SNR) remains constant.

For our rate selection scheme, we accumulate a table that nodes in a network can use to determine the optimal transmission rate for a packet. Something similar is discussed in [7] in regard to IEEE 802.11g and rate adaptation, however, there are some differences. For instance, our packet drop probability is derived from a link's capacity distribution. In addition, we determine the optimal transmission rate for payload size ranges at discrete SNR values. This is discussed further in Chapter 3. Ultimately, the idea is

to use a pre-generated table of optimal transmission rates according to both payload size and SNR at each node in a network. The goal is to maximize the overall network goodput by selecting the goodput optimal transmission rate.

2.4 The Network Simulator

The Network Simulator (NS-2) is open-source software that is used to simulate both wired and wireless networks. It provides support for wireless simulations operating under the IEEE 802.11 protocol and is primarily designed to run on UNIX and Linux operating systems. Certain versions of NS-2 may be installed on Windows and Mac OS computers, however. In order to use NS-2, a C++ compiler is required.

The parameters of a given NS-2 simulation are initialized in a Tool Command Language (TCL) file. For instance, variables such as the propagation model, antenna type, and routing protocol can be changed to configure the simulation as desired. This provides the user with flexibility in running simulations with certain wireless communication standards. As an example, IEEE 802.11g could be implemented into a simulation by setting variables like the carrier frequency to 2.4 GHz, the slot time to 9 μs, SIFS to 10 μs, PLCP header length to 48 bits, and PLCP preamble length to 144 bits (or 72 bits if using a short preamble). In addition to configuring the wireless protocol, a topology must be defined. This includes setting the number of nodes in the network and how they are positioned relative to one another. Traffic flows must also be generated so packets can be sent across the network; NS-2 supports both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) traffic. After the TCL file has been configured

entirely, NS-2 can be executed. When the simulation has completed, a trace file is generated, which records events such as when packets are sent, received, dropped, and forwarded. This file can be parsed to analyze the results of the simulation.

One drawback to NS-2 is that support for multirate wireless networks is not included in standard releases. That is, each node in the network will transmit packets with the same user-specified data rate. In order for this feature to be implemented, modifications to NS-2 are needed. Currently, there are some multirate implementations of NS-2 that have been released by other individuals and research group. For this thesis, modifications were made to the standard NS-2 version 2.34 release to incorporate our multirate MIMO scheme.

CHAPTER 3

MIMO LINK MODEL

3.1 MIMO Link Model and Creation of Capacity Distributions

To model our MIMO link, we first generate a channel gain matrix, **H**, where $H_{r,c} \sim CN(0,1)$, with r denoting the row of the matrix and c denoting the column of the matrix. In other words, each element of the matrix is an independent, identically distributed, complex Gaussian random variable with zero mean and unit variance. Each element represents the complex gain for some transmitter-receiver antenna pair. Unit variance is achieved through normalization of the channel gain matrix (i.e. the real and imaginary parts of the complex normal random variables have been normalized to have variances of $\frac{1}{2}$). Proper adjustment of the transmission power can be utilized to achieve this normalization [14]. Thus, each element of the channel gain matrix is of the form

$$H_{r,c} = \frac{1}{\sqrt{2}} X_{r,c} + \frac{j}{\sqrt{2}} Y_{r,c} \tag{1}$$

where $X_{r,c} \sim N(0,1)$ and $Y_{r,c} \sim N(0,1)$. That is, for receiver r and transmitter c the complex channel gain for the antenna pair is $H_{r,c}$. It should be noted that both $X_{r,c}$ and $Y_{r,c}$ are independent of one another.

This is known as the channel gain matrix for an uncorrelated Rayleigh fad-

ing channel and is an appropriate model for environments with large amounts of scattering (e.g. indoors). This model assumes that there is no correlation across the transmit antennas and likewise for the receive antennas. Such a model would result in a channel gain matrix with some associated covariance matrix Σ [15].

We now have a channel gain matrix that will allow us to appropriately model a Rayleigh fading channel for our MIMO link. We start with the equation

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{2}$$

For T transmit antennas and R receive antennas we have $\mathbf{y}$, a R x 1 vector of received signals at each respective receive antenna, $\mathbf{x}$, a T x 1 vector of signals transmitted, $\mathbf{n}$, a R x 1 noise vector, and $\mathbf{H}$, our R x T channel gain matrix. Further, we use a discrete time model for our channel. In other words,

$$\mathbf{y}[i] = \mathbf{H}\mathbf{x}[i] + \mathbf{n}[i] \tag{3}$$

with the values of each entry in a given vector occurring at some time-sampled index, i.

We are concerned primarily with the capacity over this link. We can use the singular value decomposition (SVD) of our channel gain matrix to transmit using $s \leq \min(T,R)$ streams [16], [17]. This allows us to break up the channel into s independent SISO subchannels [18]. We first compute the

SVD of our channel gain matrix:

$$\mathbf{H} = \mathbf{ADB}^* \tag{4}$$

with $\mathbf{A}$ and $\mathbf{B}$ being complex R x R and T x T rotation matrices respectively, $\mathbf{D}$ being a rectangular diagonal matrix, and * denoting the conjugate transpose. We then define $\mathbf{V}$ as the s most dominant right-singular vectors of $\mathbf{H}$. Our new effective channel gain matrix is known as $\mathbf{H}_{\text{eff}} = \mathbf{HV}$. From [19, p. 348] we know the capacity for a T x R Rayleigh fading MIMO channel with each antenna transmitting at equal power is

$$C = \log(\det(\mathbf{I} + \frac{SNR}{T}\mathbf{HH}^*)) \tag{5}$$

With our new channel gain matrix the capacity is now

$$\begin{aligned} C &= \log(\det(\mathbf{I} + \frac{SNR}{s}\mathbf{H}_{\text{eff}}\mathbf{H}^*{}_{\text{eff}})) \\ &= \log(\det(\mathbf{I} + \frac{SNR}{s}\mathbf{HVV}^*\mathbf{H}^*)) \text{ bits per channel use} \end{aligned} \tag{6}$$

with $\mathbf{I}$ being an R x R identity matrix. The SNR is equal to $\frac{P}{N_0}$, with P being the total signal power and N_0 being the total noise power. This capacity value is for the entire channel, or all streams, when the power is allocated evenly across the streams. Any one stream will see an SNR value of $\frac{P}{sN_0}$. The units of our capacity result, bits per channel use, reflect the fact that we are using a discrete-time model for our channel. This can be converted to continuous time units provided we know the rate at which we signal into the channel. If the bandwidth of our channel is W Hz, then we can use this as our signaling

rate conversion factor. Therefore,

$$C_{CT} = C \cdot W \text{ bits per second} \tag{7}$$

where C_{CT} is the continuous time channel capacity.

Given a link between two wireless nodes, we can generate both the PDF and CDF for the channel capacity with the following known values: the number of transmit antennas, the number of receive antennas, the signal-to-noise ratio, and the number of streams to use during transmission. Using these values, we can utilize the steps described earlier to generate our distributions. This is explained in greater detail in Figure 3.1.

PDF and CDF Generation for MIMO Link

1. Generate a channel gain matrix, $\mathbf{H}$.
2. Perform an SVD on $\mathbf{H}$.
3. Generate an effective channel gain matrix, $\mathbf{H}_{\text{eff}}$.
4. Calculate the channel capacity for the given $\mathbf{H}_{\text{eff}}$.
5. Store the capacity value.
6. Repeat for N iterations.
7. After N iterations, generate a normalized histogram of the capacity values to produce the PDF.
8. Generate the CDF using the PDF.

Figure 3.1: Algorithm to Create PDF and CDF of a MIMO Link's Channel Capacity

To illustrate, the PDF and CDF for a 4 x 4 link operating over four streams is displayed in Figure 3.2. The SNR of this particular link is set at 30 dB and the bandwidth used is 20 MHz, which is commonly associated with the IEEE 802.11 protocol.

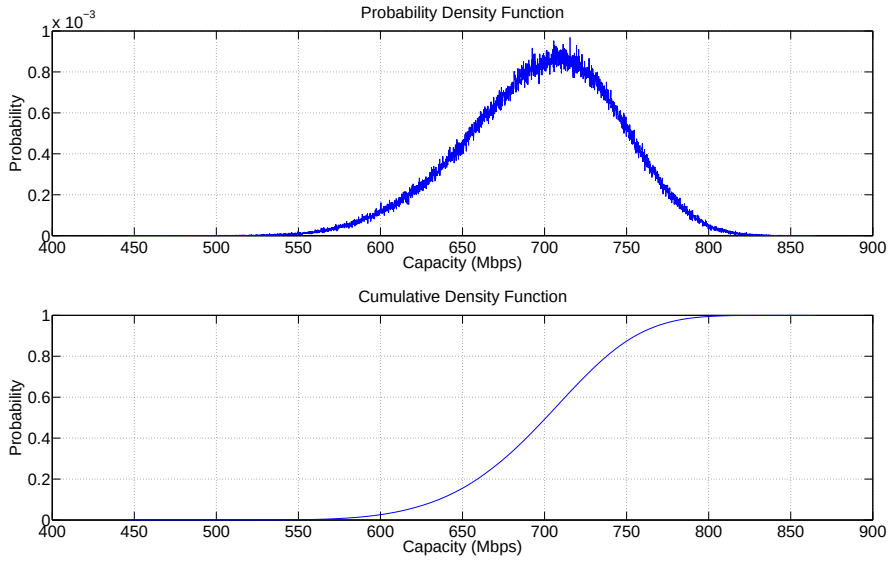

Figure 3.2: PDF and CDF of 4 x 4 MIMO Link, Four Streams, 30 dB SNR

It should be noted that the distributions generated are theoretical and may result in capacity values that are overly optimistic. This is because the values are based on the information theory capacity and do not account for certain elements such as modulation and coding scheme, or symbol rates.

3.2 Modeling Successful Packet Transmission

Once the distributions have been produced for a given link, we can use them to model successful packet transmission over that link. For example, if we choose to transmit over a link with a data rate of R_{data} bps, then we can compare that selection against the link's CDF. A certain percentage of channel capacities for that link, p_{below}, will fall below that rate, and likewise a percentage of capacities, $p_{above} = 1 - p_{below}$, will fall above that rate. We can interpret this as there being a p_{below} probability that the rate we transmitted with exceeds the channel capacity. Using this, we can say that a

packet transmitted over the link with rate R_{data} bps will be dropped with probability p_{below}. This is equivalent to saying the packet will be successfully transmitted with probability p_{above}.

Instead of selecting a transmission rate and subsequently determining its associated probability of success, another option is to determine the transmission rate for a given desired probability of success, σ. The probability a packet will be dropped is then $\epsilon = 1 - \sigma$. To determine the rate at which to transmit we simply use the CDF to select R_{data} bps such that

$$P\left[C_{CT} \leq R_{data}\right] = \epsilon \tag{8}$$

However, since most protocols have a set of fixed data rates, it is more reasonable to determine the optimal transmission rate among this set of possible rates. After generating the distribution for a link, we use a metric of our choice to compare the set of rates with one another. The goodput is the metric chosen for this purpose.

3.3 Calculating the Time to Transmit a Packet

First, the amount of time for one frame transmission is needed for our goodput calculation. Table 3.1 provides protocol-specific information in regard to values used in this chapter, which we label as WLAN MIMO in the table. In addition, the table also includes information regarding IEEE 802.11b and IEEE 802.11g for comparison purposes [6].

Table 3.1: Protocol Parameters for IEEE 802.11b, IEEE 802.11g, and WLAN MIMO

Protocol	Rates (Mbps)	Slot Time (μs)	DIFS (μs)	Total PLCP (μs)	SIFS (μs)	ACK (μs)
802.11b	1, 2, 5.5, 11	20	50	96 or 192	10	112[1]
802.11g	6, 9, 12, 18, 24, 36, 54	9 or 20	28 or 50	20, 96 or 192	10	112[1]
WLAN MIMO	400, 450, 500, 550, 600, 650, 700, 750	9	28	5	10	4.66

For the examples used in this chapter, we assume the ACK is 14 bytes long and is transmitted at a rate of 24 Mbps, which results in a total ACK time of 4.66 μs. We also assume the PLCP header is 6 bytes long and the preamble is 9 bytes, both transmitted at a rate of 24 Mbps. This gives us the 5 μs length shown in the table.

In addition to timing, when transmitting a packet, the size of the packet must be known in order to calculate the goodput. Table 3.2 shows the allocation of bytes for both the overhead and payload of an IEEE 802.11 MAC frame.

Table 3.2: Allocation of Bytes for IEEE 802.11 MAC Frame

MAC Protocol Data Unit (MPDU)			PLCP		Acknowledgment
MAC Header	Data	CRC	Preamble	Header	ACK
30	0 - 2312	4	9 or 18	6	14

Using the information we have aggregated, the total time to transmit a

[1]Timing is calculated under the assumption the ACK is 14 bytes long and transmitted at 1 Mbps.

frame will be a function of the form

$$t_{frame,R_{data}} = \tau + \frac{b_{data}}{R_{data}} + \frac{b_{ACK}}{R_{ACK}} \text{ seconds} \tag{9}$$

In other words, the total time to transmit a single frame will be the sum of a fixed amount of time, τ, which is dependent on the protocol overhead, and two variable costs: the amount of time it takes to transmit a frame (including MAC header and CRC) of b_{data} bits at rate R_{data} bps, and the amount of time it takes to transmit an ACK of b_{ACK} bits at rate R_{ACK} bps. We know that the fixed amount of time is equal to

$$\tau = DIFS + BACKOFF + 2 \cdot PLCP + SIFS \text{ seconds} \tag{10}$$

The $BACKOFF$ variable represents the backoff interval. After a node has waited for the entire $DIFS$ period, it randomly selects a number within an interval known as the contention window if the medium is busy. This interval ranges from zero to CW_{min}, where CW_{min} is the minimum contention window size. After the random number is selected, the node decrements the number every subsequent slot a transmission does not occur. When the number reaches zero, the node attempts to transmit. If a collision occurs, the contention window size is doubled and another number is selected. The maximum size for the contention window is known as CW_{max}. The $PLCP$ variable is accounted for twice because a preamble and header is transmitted once for the payload and once for the acknowledgment.

The fixed transmission time will vary depending on the protocol and its associated parameters. For the backoff time, we will use the average value

that any node will see in the absence of packet losses, which is

$$BACKOFF = \frac{CW_{min} \cdot Slot\ Time}{2} \text{ seconds} \tag{11}$$

since the values in the contention window are uniformly distributed. As an example, we can calculate the fixed transmission for WLAN MIMO using the values from our Table 3.1. Assuming the contention window size is 16, then the fixed transmission time is 120.66 μs.

As mentioned earlier, we are modeling packet reception with Bernoulli trials. Therefore, if the probability of successful packet transmission over link l at rate R_{data} bps is $p_{l,R_{data}}$ then the expected number of transmission attempts until success is simply the mean of a geometric distribution with parameter $p_{l,R_{data}}$. This is just $\frac{1}{p_{l,R_{data}}}$. With this, we can approximate the total time to transmit a frame over a link at rate R_{data} bps:

$$\begin{aligned} t_{total,R_{data}} &= \frac{t_{frame,R_{data}}}{p_{l,R_{data}}} \\ &= \frac{\tau + \frac{b_{data}}{R_{data}} + \frac{b_{ACK}}{R_{ACK}}}{p_{l,R_{data}}} \text{ seconds} \end{aligned} \tag{12}$$

The result is a linear function of payload size (in bits); for each rate, we can plot this function for a given protocol. Figure 3.3 is a plot using the timing values of WLAN MIMO from Table 3.1. In addition, the minimum contention window size used is 16. The rates used, 400 Mbps to 750 Mbps in increments of 50 Mbps, are also taken from Table 3.1.

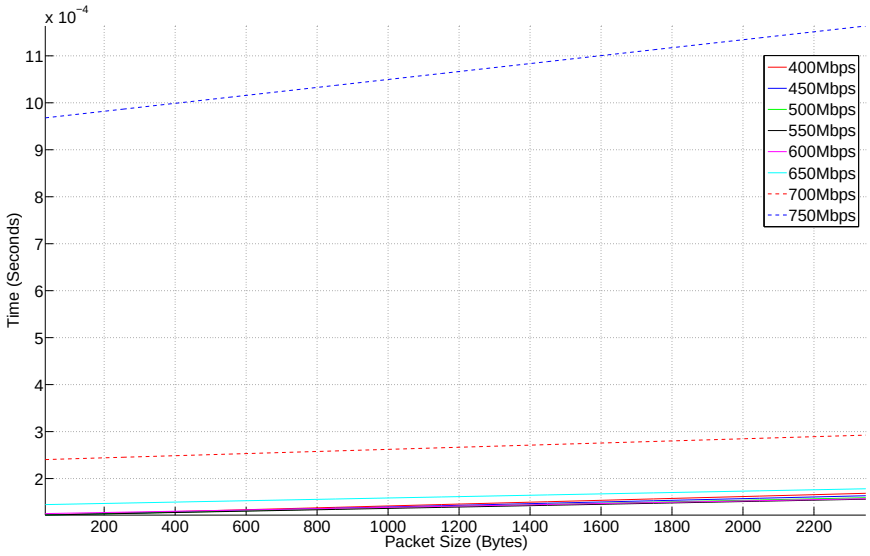

Figure 3.3: Transmission Time for 4 x 4 MIMO Link, Four Streams, 30 dB

Figure 3.3 corresponds to the 4 x 4, four stream, 30 dB SNR link with PDF and CDF generated in Figure 3.2. That is, the total time to transmit the payload for each rate was computed using probabilities extracted from the link's distribution. In Figure 3.3 it is clear that the 750 Mbps rate will take the longest time to transmit a packet for all sizes. The 700 Mbps rate also takes much longer to transmit a packet in comparison to the remaining rates. This is because of their probability of successful packet transmission. Both rates have low probabilities of success, which is apparent from the location of these values in Figure 3.2. The associated probabilities for each rate can be found in Table 3.3.

Table 3.3: Probability of Successful Transmission for a Single Packet

400 Mbps	**450 Mbps**	**500 Mbps**	**550 Mbps**	**600 Mbps**	**650 Mbps**	**700 Mbps**	**750 Mbps**
1.000000	0.999999	0.999968	0.998381	0.973827	0.844655	0.507738	0.126072

3.4 Calculation of Goodput

To calculate the goodput, we simply divide the number of bits in the payload (i.e. we do not include the MAC header and CRC) by the total amount of transmission time:

$$\begin{aligned} g_{R_{data}} &= \frac{b_{payload}}{t_{total,R_{data}}} \\ &= \frac{p_{l,R_{data}} \cdot b_{payload}}{\tau + \frac{b_{data}}{R_{data}} + \frac{b_{ACK}}{R_{ACK}}} \text{ bits per second} \end{aligned} \tag{13}$$

This function can also be plotted to see the difference in curves among the different rates. Figure 3.4 contains the associated goodput plot for the rates in Figure 3.3. From Figure 3.4, it can be seen that the 700 Mbps and 750 Mbps rates experience a large drop in goodput relative to the other transmission rates. This can be attributed to their low probability of successful packet transmission. The 650 Mbps rate experiences a similar drop in goodput, but not to the extent of the previously mentioned rates.

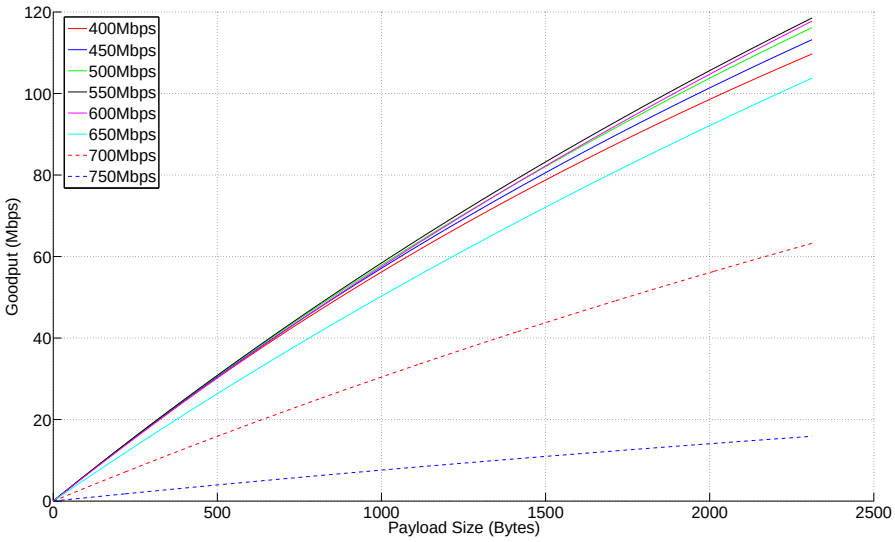

Figure 3.4: Goodput for 4 x 4 MIMO Link, Four Streams, 30 dB

After calculating the goodput for each rate, the optimal rate of transmission can then be determined for a set of ranges according to payload size. For this particular example, the ranges are listed in Table 3.4.

Table 3.4: Optimal Transmission Rate for WLAN MIMO, 4 x 4, Four Streams, 30 dB

Optimal Rate (Mbps)	**Payload Range (Bytes)**
500	1 - 102
550	103 - 2312

Out of the eight possible rates to transmit with, only two are used: 500 Mbps and 550 Mbps. Neither of these rates is the fastest possible, yet they are optimal for the given ranges. This can be viewed as a decision based on a tradeoff. It is not worth transmitting at one of the larger rates due to the number of packets that will be lost. However, even though 500 Mbps and 550 Mbps have lower associated probabilities than 400 Mbps or 450 Mbps, the

difference is nearly negligible. In this case, it makes more sense to transmit at the rates of 500 Mbps or 550 Mbps. This tradeoff is of course based on our goodput calculations.

Figure 3.5 has been provided to demonstrate the intuition behind our rate selection algorithm. We use the information from our previous plots and tables. From Table 3.4, we can see for this link we have two distinct payload ranges. The median payload size for each of these ranges is obtained and used to plot goodput over the set of transmission rates. For example, the median payload size for the first range, 1 to 102 bytes, is 52 bytes, while the median payload size for the second range, 103 to 2312 bytes, is 1207 bytes. We take these two payload sizes, calculate the goodput for each of our transmission rates, and plot our results in Figure 3.5.

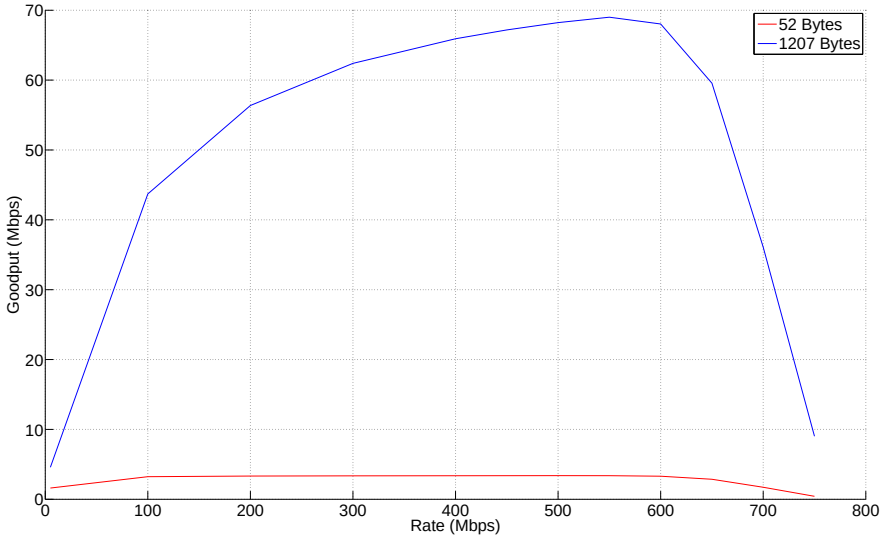

Figure 3.5: Goodput of Median Range Values of 4 x 4 MIMO Link, Four Streams, 30 dB SNR

From Figure 3.5, we can make conclusions about the effect of payload size in our method of rate selection. For small payload sizes, the rate at which the packet is transmitted does not have much influence on the goodput. For

instance, the 52 byte payload has a relatively flat goodput measurement. However, for larger sizes, such as the 1207 byte payload, the goodput varies more with changes to the transmission rate. Intuitively, this makes sense. The fixed transmission time will be the dominant term in the total transmission time for smaller payloads, causing the transmission rate to have less influence on the goodput. For larger payloads, the variable transmission time dominates. As a result, the transmission rate needs to be selected more carefully because of the goodput's heightened sensitivity to increases or decreases in rate.

CHAPTER 4

NS-2 MODIFICATIONS

NS-2 [20] is open-source software that is used to simulate both wired and wireless networks. It provides support for wireless simulations operating under the 802.11 protocol. However, it does not include an implementation of wireless networks with MIMO links. By integrating MATLAB with NS-2, we incorporate our MIMO link model into our NS-2 simulations. The algorithm to generate a single link's distributions was programmed using MATLAB. This is the same algorithm from Figure 3.1. The flow of execution to incorporate this MATLAB code into NS-2 is displayed in Figure 4.1. The dotted lines represent input or output while the solid lines represent a flow of execution from one module to another.

First, the user must specify the following values: the number of transmit antennas, the number of receive antennas, the number of streams to use, a set of transmission rates, a set of SNR values, and other timing parameter values. For simplification, the links in the simulation are assumed to be homogenous in regard to their parameters; transmissions across each link operate with the same number of transmit antennas, receive antennas, and over the same number of streams. With this information, the program can generate a table of optimal rates for payload ranges, according to the SNR values and rates the user specified.

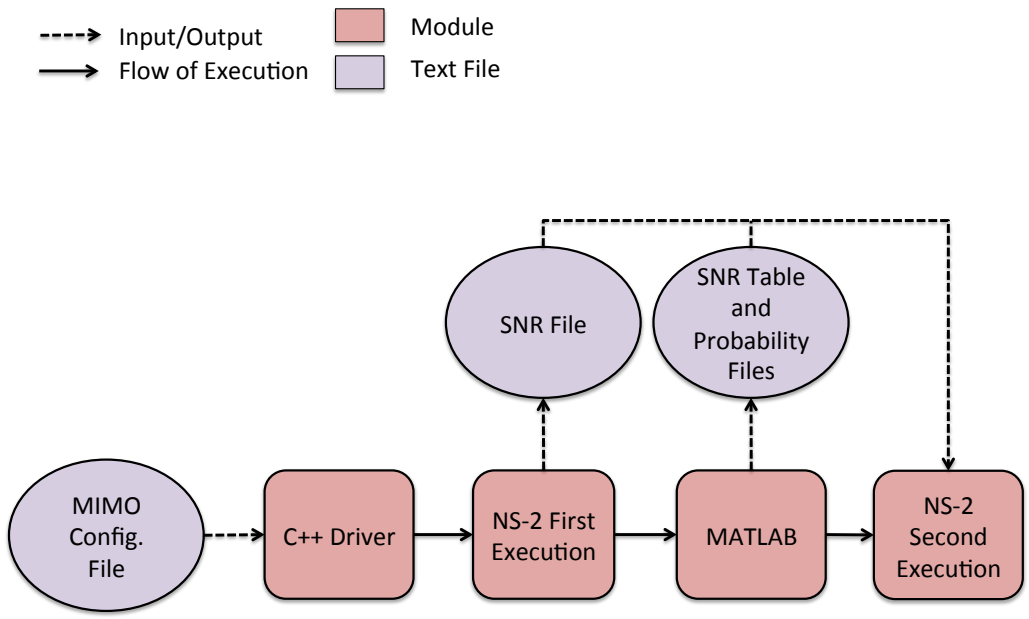

Figure 4.1: Flow Chart for Integration of NS-2 and MATLAB

As mentioned in Section 3.1, to generate the PDF and CDF for a link, one of the values needed is the SNR. This requires knowledge about the power level of packets received over the links. Therefore, for a given topology and set of traffic flows, two executions of an NS-2 simulation file are needed. The first execution produces the SNR values for each of the links used during the simulation. In addition, such an implementation only works for static topologies; if the nodes were mobile, distances across links would change, resulting in multiple SNR values for a link and multiple distributions depending on the current positions of the nodes.

After the first simulation run has completed and the table has been generated, we execute the simulation one final time; however, during the final run we use the values in our table to choose the optimal transmission rate over

each link. When a packet is to be sent across a link, we first obtain the SNR over the link (which was calculated during the first simulation run) and the payload size of the packet. We then look up the closest SNR in our table, and what range the payload belongs to. From this, we obtain the optimal rate to transmit the packet with and set the `dataRate_` variable in NS-2 to the appropriate value.

4.1 Capturing Link SNR Values

NS-2 offers three propagation models that can be used in wireless simulations: free space, two-ray ground, and shadowing. These models are used to estimate, mathematically, the signal strength at a receiver based on a set of transmitter, receiver, and environment parameters. The environment that transmission takes place in determines which propagation model is most appropriate for signal strength estimation. For example, the free space propagation model is accurate when there are no obstructions between transmitter and receiver. As its name suggests, the two-ray ground reflection model is best for environments in which both the direct signal path as well as the reflected signal from the ground need to be taken into account for signal loss. Finally, shadowing takes into account the loss of signal strength due to reflection, scattering, and absorption when an obstruction is in the direct path of the signal. For simulations run in this thesis, the two-ray ground reflection model is used. The equation to compute the received signal power using this model is

$$P_r = \frac{P_t \cdot G_t \cdot G_r \cdot h_t^2 \cdot h_r^2}{d^4 \cdot L} \text{ Watts} \tag{14}$$

where P_r is the received signal power, P_t is the transmission power, G_t is the transmit antenna gain, G_r is the receive antenna gain, h_t is the height of the transmit antenna, h_r is the height of the receive antenna, d is the distance between the transmit and receive antenna, and L is the system loss. In addition, because the two-ray ground model is not accurate in short distances, NS-2 uses a cross-over distance threshold

$$d_c = \frac{4 \cdot \pi \cdot h_t \cdot h_r}{\lambda} \text{ meters} \tag{15}$$

where λ is the wavelength, to improve accuracy. If the distance between the transmitter and receiver is above the cross-over distance, then the two-ray ground model, Equation (14), is used. However, if the distance falls below the threshold then the free space model

$$P_r = \frac{P_t \cdot G_t \cdot G_r \cdot \lambda^2}{(4\pi)^2 \cdot d^2 \cdot L} \text{ Watts} \tag{16}$$

is used [20, p. 190]. It should be noted that this equation does not hold for $d \to 0$.

The first modification to NS-2 is made with the goal of capturing the SNR between each link that is used during the simulation. First, three variables are added to the `tworayground.h` header file: `noisePwr_`, `numNodes_`, and `linkVector`. The first two variables must be initialized in the TCL simulation file by the user. Figure 4.2 provides an example of how the two variables are initialized.

Noise Power and Number of Nodes Initialization in TCL File

```
...

Propagation/TwoRayGround set noisePwr_ 8.007e-14

Propagation/TwoRayGround set numNodes_ 20

...
```

Figure 4.2: Initializing Noise Power and Number of Nodes in TCL File

The `noisePwr_` variable is a constant that represents the noise power (in Watts), and `numNodes_` must be initialized to the number of nodes being used in the simulation. This allows for the following modifications, shown in Figure 4.3, to the `TwoRayGround` constructor. The first two lines simply bind the user input to the variables. The third line initializes each element of a vector to zero. This vector is used to keep track of which links have had their SNR value recorded. If the link's value in `linkVector` is one, the SNR of the link has already been recorded and does not need to be captured again. These steps are shown in Figure 4.4.

TwoRayGround Constructor Modification

```
TwoRayGround::TwoRayGround()
{

     ...

     bind("noisePwr_", noisePwr_);
     bind("numNodes_", numNodes_);
     linkVector.assign(numNodes_*numNodes_,0);

     ...

}
```

Figure 4.3: TwoRayGround Constructor Modification

Every time a packet is transmitted between two nodes under the two-ray ground model, the `Pr` function from the `tworayground.cc` source file is called

to calculate the received power. In this function, a modification is made to capture the SNR value of the link. The pseudocode in Figure 4.4 shows how this is done.

Algorithm 1 Pseudocode to Capture SNR Values

if using MIMO links && first simulation run **then**
  if link value in linkVector == 1 **then**
    • Do nothing, link SNR has already been recorded
  else
    • Calculate link's SNR
    • Record transmit node number, receive node number, and SNR to 'snrvalues.txt' output file
    • Set link value in linkVector = 1
  end if
end if

Figure 4.4: Pseudocode to Capture SNR Values

The code first checks if specific text files exist to determine whether the simulation is being run with MIMO links, and if it is, whether or not it is the first simulation run. In order to run NS-2 with MIMO links, a configuration file must be created; this configuration file will be discussed in Section 4.2. In addition, after the MATLAB code has generated the SNR table to be used in the second simulation run, it creates two text files. If these files exist, then NS-2 knows that the simulation is in its second execution and there is no need to record SNR values. This prevents unnecessarily recording the SNR data twice and lengthening the execution time.

4.2 Determining Optimal Transmission Rate across Links

After the first execution of the simulation has completed and the SNR values of the links have been captured, the MATLAB code then begins to execute. The MATLAB function requires twelve arguments: the number of transmit antennas, the number of receive antennas, the number of streams to use, a set of SNR values for the table, a set of transmission rates, the SIFS length, the slot time of the protocol being used, the basic rate for control packets, the contention window size, the number of PLCP header bits, the PLCP header rate, and the number of preamble bits. Each of these values is retrieved from the configuration file `mimoconfig.txt` mentioned previously. The driver retrieves these values and passes them to the MATLAB function when it is time to generate the table. For example, Figure 4.5 shows a sample configuration file. The first three lines of the configuration specify the path to the `ns` file, which allows the driver to run NS-2, the simulation file to use, and the location of MATLAB on the local computer. In order for the driver to function properly this information must be provided. The remainder of the configuration file contains the parameters needed for the MATLAB code. The SNR values are in units of dB, the rates in Mbps, and the SIFS and slot time in μs.

Sample MIMO Configuration File

```
NS=/path/to/ns
SIMFILE=/path/to/simulation/file
MATLAB=/path/to/MATLAB_R2012b.app
NUM_TRANSMIT=4
NUM_RECEIVE=4
NUM_STREAMS=4
SNR_VALUES=10,20,30
RATES=400,450,500,600,650,700,750
SIFS=10
SLOT=9
BASICRATE=24
CW=16
PLCP_BITS=48
PLCP_RATE=24
PREAMBLE_BITS=72
```

Figure 4.5: Sample MIMO Configuration File

For each SNR value specified, the MATLAB code generates a capacity distribution, determines the probability of success for each rate, and determines the optimal transmission rate based on payload size. When the code has finished running, it generates two output files: `SNRTable.txt` and `RateProb.txt`. The first file contains payload ranges and their associated optimal rate for each SNR value. The second file contains the transmission rates and their probability of success at each SNR value.

With this information available the next modification to NS-2 is possible. This is done in the `mac-802_11.cc` source file and the `mac-802_11.h` header file. First, four vector variables are added to the header file: `snrVector`, `rangeVector`, `rateVector`, and `probVector`. The first two vectors contain the table of SNR values and optimal rates for ranges of payload size. The last two vectors contain the transmission rates and their probability of success at each SNR.

In the source file's constructor, code is added to parse the two text files and store the information in the appropriate vector. This allows the vectors to be

scanned in the source file's `sendData` function in order to select the optimal transmission rate. This function is responsible for preparing data packets for transmission and attaching an appropriate MAC header. Pseudocode is provided in Figure 4.6 to show how the optimal rate is selected.

Algorithm 2 Pseudocode to Select Optimal Rate

if using MIMO links && second simulation run **then**
  if packet is not a broadcast packet **then**
    - Determine which link packet is being sent over and scan snrvalues.txt to get link's actual SNR
    - Scan snrVector and round the link's actual SNR to the nearest SNR value that was listed in mimoconfig.txt
    - Determine payload size of packet to be sent
    - Scan rangeVector for ranges according to selected SNR and get the optimal rate
    - Use rateVector and probVector to get the probability of success for the given rate and SNR
    - Stamp the probability of success to the packet
    - Set dataRate_ variable to optimal rate
  end if
end if

Figure 4.6: Pseudocode to Select Optimal Rate

The probability of success is attached to the packet so that it is used later to determine whether or not the packet is dropped. This is possible by adding a `success_prob` variable to the `packet-stamp.h` header file. In addition, two functions, `getProb` and `setProb`, are added to allow for access and modification of the variable.

The method for selecting which SNR in the table to use is simple: use the SNR in the table that is closest to the link's actual SNR. The accuracy of this selection method is dependent on how many SNR values the user lists in `mimoconfig.txt`, as well as the separation between the values. This introduces a particular tradeoff. Providing several, properly spaced SNR

values will result in a more accurate simulation, but increase execution time in order to generate distributions for each value. Reducing the number of values will minimize the execution time, but will result in a less accurate model. It is left up to the user to decide which and how many SNR values are used.

4.3 Determining If a Received Packet Should Be Dropped

The final major modification to NS-2 is made in the `recv_timer` function, which is responsible for packet reception. All that needs to be added is code that determines whether the packet should to be dropped based on its probability of successful transmission. Pseudocode is provided in Figure 4.7 to show how this is done.

Algorithm 3 Pseudocode to Drop a Packet

```
if using MIMO links && second simulation run then
    if receiving node == destination && packet is a data packet then
        • Generate a random number between 0 and 1
        • Get probability of successful transmission from packet stamp
        if random number is greater than probability of success then
            • Drop the packet
        else
            • Do not drop the packet
        end if
    end if
end if
```

Figure 4.7: Pseudocode to Drop a Packet

This modification does not require any changes to how NS-2 already implements the reception of packets. After the physical layer performs its threshold

checks (if the received packet power is above the capture threshold as well as the receive threshold), it determines whether or not to send the packet to the MAC layer. If the packet is sent up to the MAC layer, the receiving node first does a silent discard if it was in transmit mode during reception. The node then performs the steps in the pseudocode in Figure 4.7. If the packet has still not been dropped or discarded at this point, the node checks if a collision has occurred, then determines whether or not the packet should be dropped due to too many bit errors. The remainder of the function consists of updating the Network Allocation Vector (NAV), collecting neighbor information, address filtering, and handling the packet according to its type (broadcast, control, data, etc.).

CHAPTER 5

SIMULATION RESULTS

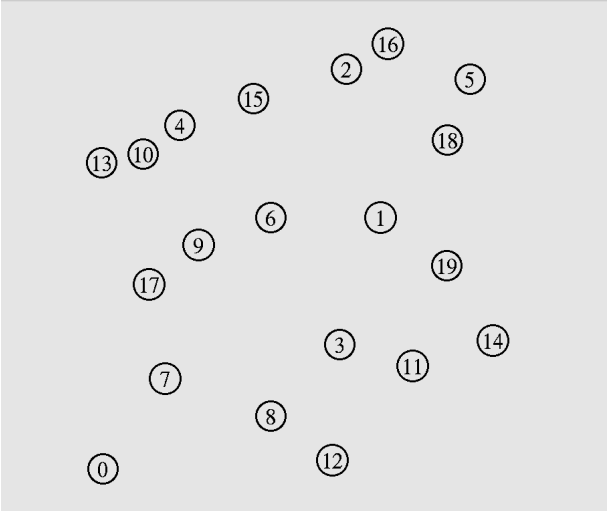

Figure 5.1: Simulation Topology

Figure 5.1 is the topology used in simulations to test our rate selection algorithm. The topology consists of 20 nodes contained in a 150 x 150 meter area. The transmission power for all nodes is set to 1 mW. In addition, the noise power is set to -65 dBm and the routing protocol used is ad hoc on-demand distance vector routing. The following parameters are used for timing purposes when running our simulations: a SIFS of 10 μs, a slot time of 9 μs, a minimum contention window size of 16, a preamble length of 9 bytes, a PLCP header length of 6 bytes, and a basic rate and PLCP rate of 24 Mbps. We test a total of six configurations, with five trials conducted for each.

5.1 Configuration 1: Average Network Goodput for Three Flows

For our first configuration, we define three UDP traffic flows: node 1 to node 0, node 2 to node 12, and node 5 to node 7. For each flow, the packet size has been set to 2000 bytes with an interval length between transmissions of 0.2 ms. Each flow is configured to send no more than 10000 packets. The carrier sensing threshold is set to -68 dBm.

For simplification, we will refer to our rate selection algorithm from Chapter 3 as WLAN MIMO for the remainder of the chapter. For WLAN MIMO we use rates of 100 Mbps to 700 Mbps in increments of 50 Mbps. We also use SNR values of 15 dB to 50 dB in increments of 5 dB for the SNR table. Figure 5.2 shows the results obtained for the first configuration.

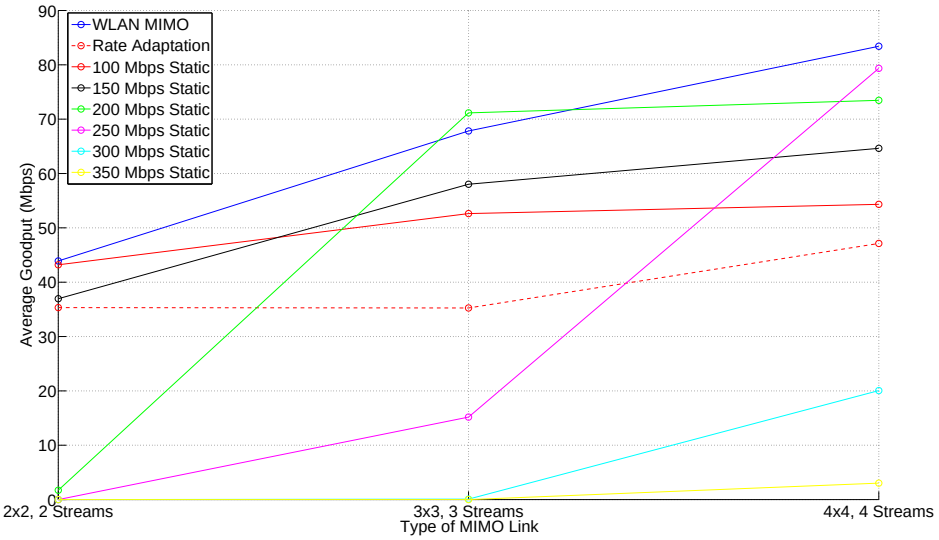

Figure 5.2: Average Network Goodput for Configuration 1

Figure 5.2 contains average network goodput measurements for three algorithms: our WLAN MIMO rate selection algorithm, a rate adaptation algorithm, and a static algorithm. We define network goodput as the total

number of payload bits that reached their destination over the course of the simulation divided by the length of time from the start of the first flow to the end of the last flow. We average this value over five trials to obtain the average network goodput for the configuration.

The rate adaptation algorithm starts by using our rate selection method described in Chapter 3 to select the goodput optimal rate. If a node completes five consecutive successful packet transmissions, it increases its transmission rate to the next highest rate, going no higher than the maximum rate, 700 Mbps. However, if the node experiences a single failed transmission, its rate reverts back to the goodput optimal rate; its transmission rate never falls below the goodput optimal rate. The static algorithm simply sets a fixed transmission rate for all nodes in the network. We test the static algorithm for each of the available rates; any rate that is not displayed in the plot is omitted because it results in an average network goodput of zero for each type of MIMO link. The three test cases are for 2 x 2 MIMO links, 3 x 3 MIMO links, and 4 x 4 MIMO links, each operating on all streams.

Figure 5.2 shows that in comparison, our rate selection algorithm performs well in terms of average network goodput for this configuration. In both the 2 x 2 and 4 x 4 MIMO link cases, it outperforms each of the other rate algorithms. For 3 x 3 MIMO links, the 200 Mbps static algorithm provides an average network goodput of 71.158 Mbps, which is higher than the 67.822 Mbps goodput WLAN MIMO produces. However, even though WLAN MIMO is not the best performing algorithm when using 2 x 2 links, it results in the second-best average network goodput. Figure 5.3 displays the average network goodput for WLAN MIMO, the rate adaptation algorithm, and the 200 Mbps static algorithm for each type of MIMO link along with one confidence interval.

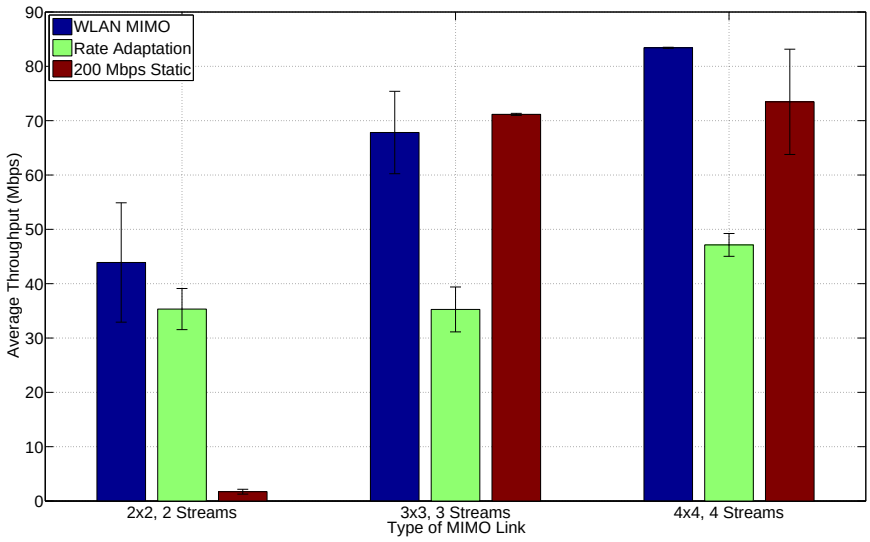

Figure 5.3: Average Network Goodput for Configuration 1 with 68% Confidence Interval

5.2 Configuration 2: Average Network Goodput for Reduced Packet Size

Five trials are run for a second configuration using the same parameters as configuration 1. This means that the same SNR tables generated from the first configuration are used as well. The only change made is to the size of packets in the UDP flows. Each flow has their packet size changed from 2000 bytes to 512 bytes. We display our results in Figure 5.4.

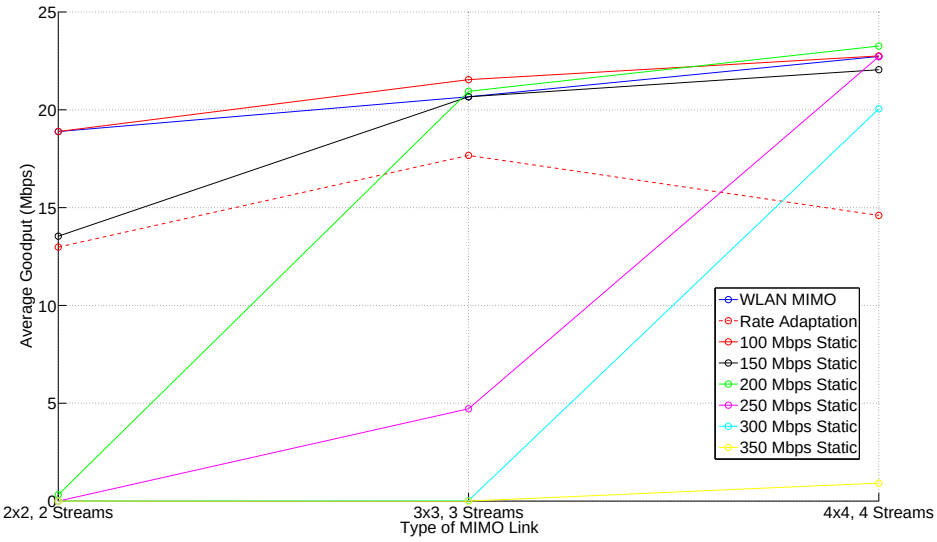

Figure 5.4: Average Network Goodput for Configuration 2

For the most part, the shapes of the curves remain fairly consistent with respect to configuration 1. The most noticeable change is for the rate adaptation algorithm, which suffers a drop in goodput going from using 3 x 3 links to using 4 x 4 links. It should be noted that although our rate selection algorithm ties for the highest goodput for 2 x 2 links, it is inferior for 3 x 3 and 4 x 4 links. For the 2 x 2 case, WLAN MIMO produces the same average goodput as the 100 Mbps static algorithm. In the 3 x 3 case, it produces the same goodput as the 150 Mbps static algorithm, but both perform worse than the 100 and 200 Mbps algorithms. These two static algorithms also outperform WLAN MIMO when using 4 x 4 links. The results from Figure 5.4 support the conclusions we made in Chapter 3 using Figure 3.5 in regard to rate selection. With a smaller packet size, the difference in goodput produced among the available rates is minimal. This applies only to rates with high associated probabilities of successful transmission. Comparing the variation in goodput in Figure 5.4 with Figure 5.2 reinforces this idea. We

include Figure 5.5 to display the confidence interval for the relevant rates of this configuration.

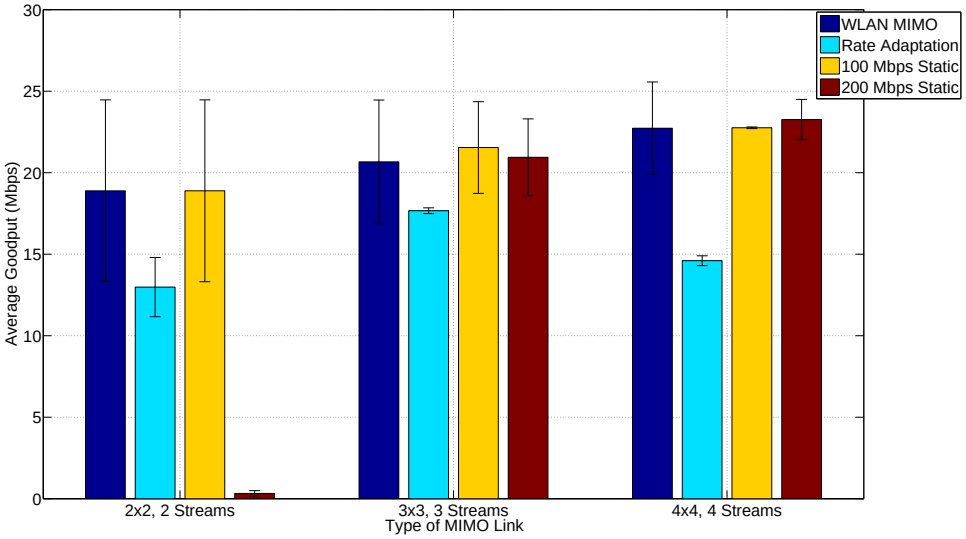

Figure 5.5: Average Network Goodput for Configuration 2 with 68% Confidence Interval

5.3 Configuration 3: Average Network Goodput for Three Different Flows

We now test to see if changing the sources and destinations of our flows has any effect on the performance of our algorithm. For the next configuration, we use the same configuration and flow parameters from configuration 1. The flows are now from node 3 to node 13, node 7 to node 19, and node 0 to node 16.

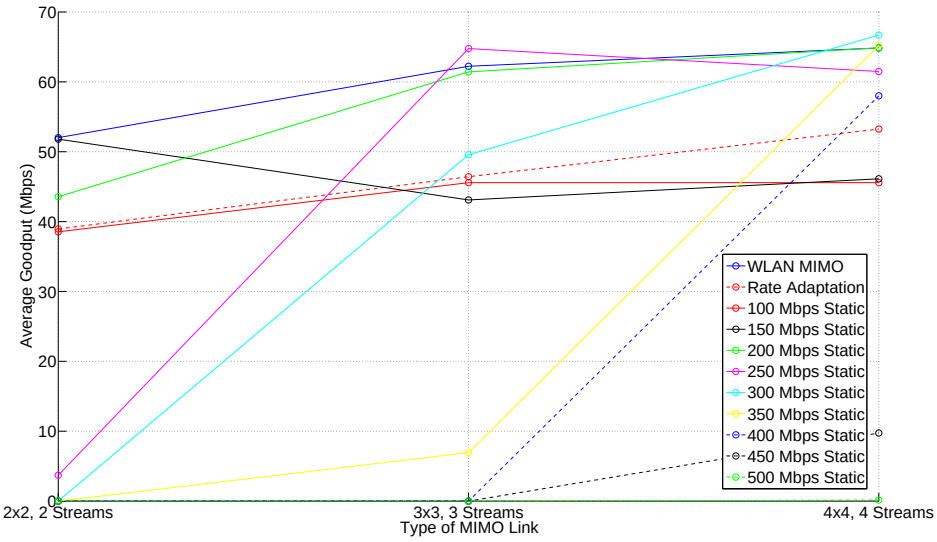

Figure 5.6: Average Network Goodput for Configuration 3

WLAN MIMO is not the best for 3 x 3 and 4 x 4 MIMO links under configuration 3, but provides the highest goodput using 2 x 2 MIMO links. This differs from our results when using configuration 1. Although we use the same parameters, WLAN MIMO does not perform as well after changing the sources and destinations of our flows. Figure 5.7 provides a more detailed comparison of the relevant rates for configuration 3.

For our final three configurations, we test the effect of routing and medium access control. For our previous three configurations, each flow could use multiple hops in its route. In addition, the carrier sensing threshold value remained the same for each configuration.

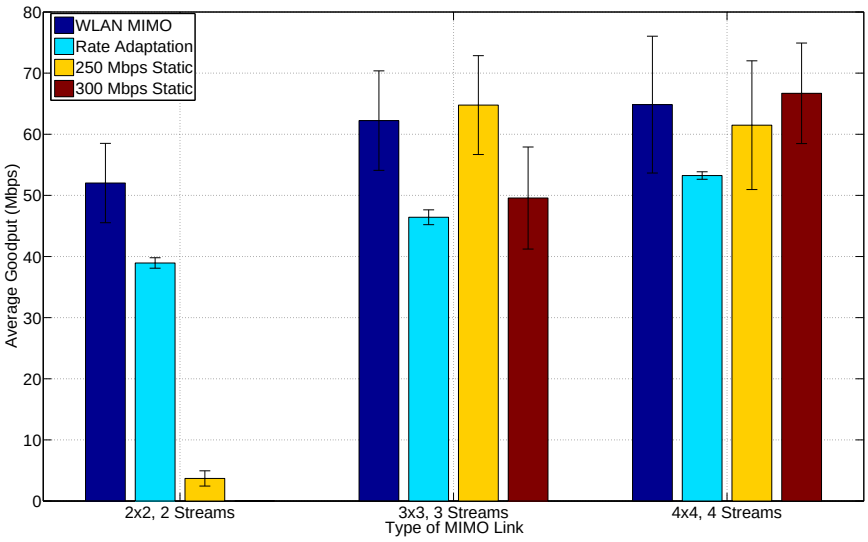

Figure 5.7: Average Network Goodput for Configuration 3 with 68% Confidence Interval

5.4 Final Three Configurations and Comparison of Configurations 1 - 6

To test the effects of these parameters, we run additional simulations with multiple single-hop flows and varying carrier sensing thresholds. Table 5.1 lists details for the three additional configurations that we use in our simulations. In addition, entries for the first three configurations that were used are also included in the table.

Table 5.1: Configuration Parameters for NS-2 Simulations

Configuration[1]	**Flows (Source, Destination)**	**Packet Size (Bytes)**	**CS Threshold (dBm)**	**Multihop**
1	(1,0) (2,12) (5,7)	2000	−68	Yes
2	(1,0) (2,12) (5,7)	512	−68	Yes
3	(3,13) (7,19) (0,16)	2000	−68	Yes
4	(1,17) (9,8)	2000	−82	No
5	(1,19) (9,17) (8,3) (11,14) (0,7)	2000	−52	No
6	(1,19) (9,17) (8,3) (11,14) (0,7)	2000	−82	No

We first look at the performance of each algorithm for each configuration when using 2 x 2 MIMO links. Figure 5.8 displays the results obtained from our simulations. The horizontal axis represents the six configurations listed in Table 5.1. For each configuration, the average network goodput from WLAN MIMO, the rate adaptation algorithm, and the static algorithms are plotted. For each configuration, WLAN MIMO is the leftmost blue bar and 700 Mbps is the rightmost red bar.

[1]Each configuration uses the same topology, AODV protocol, a 1 mW transmit power, a 0.2 ms packet interval, a maximum packet amount per flow of 10000, and −65 dBm noise power. Timing parameters are taken from Table 3.1.

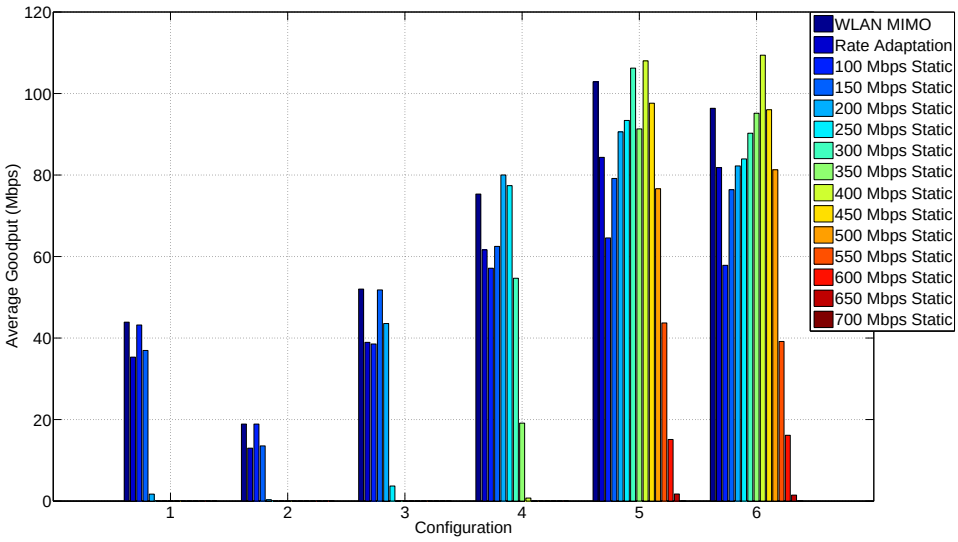

Figure 5.8: Average Network Goodput for Configurations 1 - 6 Using 2 x 2 Links

Our WLAN MIMO rate selection algorithm produces the highest goodput for configurations 1, 2, and 3 when using 2 x 2 MIMO links. For configuration 2, it ties with the 100 Mbps static algorithm. WLAN MIMO does not perform as well for configurations 4, 5, and 6, however. Although it produces the second-best goodput under configuration 6, it is still significantly less than the goodput of the best algorithm, 400 Mbps. Similar results can be seen for configurations 4 and 5, where WLAN MIMO performs worse than some of the static algorithms. Comparing the results from configurations 5 and 6, we can see that decreasing the carrier sensing threshold to -82 dBm in configuration 6 has an adverse effect on WLAN MIMO and most of the static rates, with a few exceptions. We now compare the six configurations for 3 x 3 MIMO links.

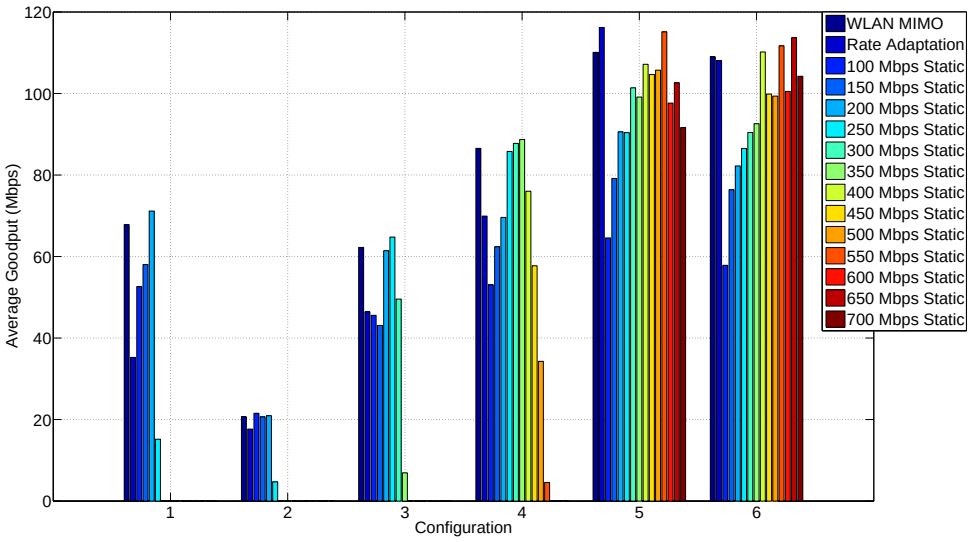

Figure 5.9: Average Network Goodput for Configurations 1 - 6 Using 3 x 3 Links

Using 3 x 3 links, WLAN MIMO is not the best performing algorithm for each of the configurations. In configuration 1, it is outperformed by 200 Mbps, and in configuration 2 by both 100 and 200 Mbps. It performs slightly better in configuration 3 in which only one other algorithm, 250 Mbps, produces a larger goodput. Viewing the results for configuration 5, it can be seen that WLAN MIMO is outperformed by the rate adaptation algorithm for the first time in simulations run thus far. Decreasing the carrier sensing threshold from -52 dBm in configuration 5 to -82 dBm in configuration 6 produces results similar to that of the 2 x 2 MIMO link case. It does appear that this decrease in threshold has more of an effect on lower static rates than the larger static rates. The final comparison we make is for 4 x 4 MIMO links.

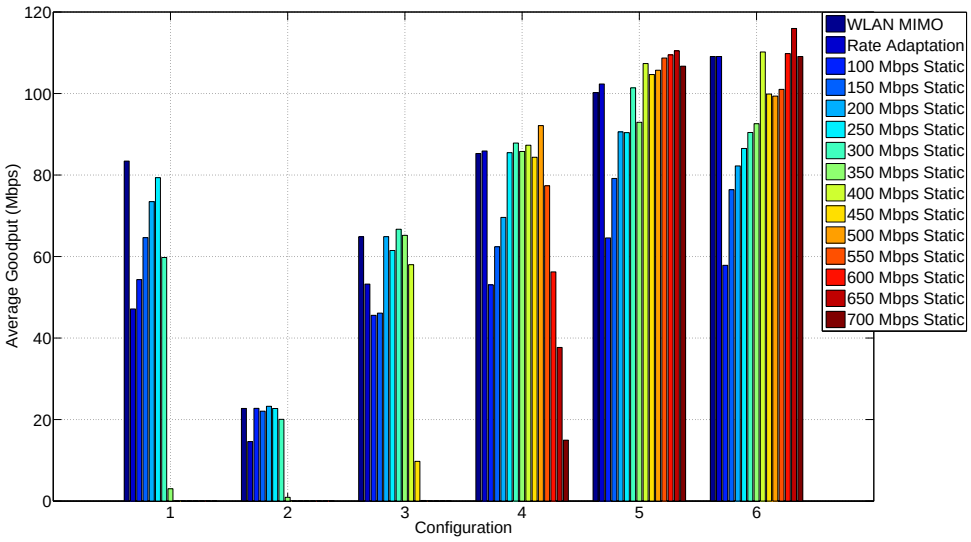

Figure 5.10: Average Network Goodput for Configurations 1 - 6 Using 4 x 4 Links

When using 4 x 4 MIMO links, WLAN MIMO is clearly the best algorithm for configuration 1 by a large margin. This margin disappears in configuration 2, with both the 100 and 200 Mbps static rates outperforming WLAN MIMO. This can be attributed to the decrease in packet size, which is the only difference between the two configurations. In configurations 5 and 6 WLAN MIMO is outperformed by several static rates. In both of these configurations, the 650 Mbps static rate is the best algorithm with WLAN MIMO producing goodput values similar to that of the rate adaptation algorithm.

5.5 Observations

From the results of our simulations, we have observed that our rate selection algorithm is the best under certain conditions and below the best scheme under other conditions. In general, WLAN MIMO performs fairly consistently across the different configurations for each type of link. In some instances

it is the best, but for the configurations in which it is not, it typically produces results that are close to the best-performing scheme. Figures 5.8, 5.9, and 5.10 show that when using 3 x 3 and 4 x 4 MIMO links, which have higher capacities, the larger static rates begin to perform better, specifically for configurations 4, 5, and 6. Decreasing the carrier sensing threshold has an adverse effect on WLAN MIMO for 2 x 2 and 3 x 3 MIMO links; when using 4 x 4 MIMO links, the opposite occurs, with WLAN MIMO performing better with a decreased carrier sensing threshold.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this thesis, we have defined a model for a MIMO link and used our model to create a rate selection algorithm (called WLAN MIMO in the thesis) that chooses the optimal transmission rate through the use of a goodput measurement. Modifications were made to NS-2 to allow for us to incorporate MIMO links and our rate selection algorithm into simulations to obtain experimental results. From our simulations, we observed that our rate selection was the best under certain conditions and below the best scheme under other conditions. In general, it performed better for configurations 1, 2, 3, and 4 for each type of MIMO link. For configurations 5 and 6, there was greater variation in the results. When using 3 x 3 and 4 x 4 MIMO links, the larger static rates began to produce more competitive goodput results, specifically for configurations 4, 5 and 6. However, overall our rate selection algorithm provides goodput values that are the best or near the best value across configurations and link types.

There are a number of modifications and additions that can be made to expand upon the work in this thesis. The capacity and rate values that we have been using are ideal. Modulation, coding schemes, and symbol rates could be factored into our model to obtain more reasonable capacity values for our links. Additional elements could also be taken into account in our rate selection algorithm in order to achieve better performance, particularly with 3 x 3 and 4 x 4 MIMO links. It would also be beneficial to incorporate

mobility into the modifications we made to NS-2 to allow for flexibility in running simulations. Additional experiments could then be conducted to study the impact of factors such as mobility, network size, and the number of competing flows on our rate selection algorithm.

REFERENCES

[1] J. T. Wang, "Throughput-based switching between diversity and multiplexing in MIMO systems," in *TENCON 2007 - 2007 IEEE Region 10 Conference*, Taipei, Taiwan, Oct. 2007, pp. 1–3.

[2] H. Bolcskei, D. Gesbert, and A. Paulraj, "On the capacity of OFDM-based spatial multiplexing systems," *Communications, IEEE Transactions on*, vol. 50, no. 2, pp. 225–234, Feb. 2002.

[3] M. El-Hajjar and L. Hanzo, "Multifunctional MIMO systems: A combined diversity and multiplexing design perspective," *Wireless Communications, IEEE*, vol. 17, pp. 73–79, Apr. 2010.

[4] A. Lozano, "Transmit diversity vs. spatial multiplexing in modern MIMO systems," *Wireless Communications, IEEE Transactions on*, vol. 9, pp. 186–197, Jan. 2010.

[5] B. P. Crow, I. Widjaja, J. G. Kim, and P. Sakai, "Investigation of the IEEE 802.11 medium access control (MAC) sublayer functions," in *Proceedings of IEEE INFOCOM 1997*, Kobe, Japan, Apr. 1997, pp. 126–133.

[6] A. Duda, "Understanding the performance of 802.11 networks," in *IEEE Personal, Indoor and Mobile Radio Communications*, Cannes, France, Sep. 2008, pp. 1–6.

[7] R. Kumar, S. Tati, F. Mello, S. Krishnamurthy, and T. Porta, "Network coding aware rate selection in multi-rate IEEE 802.11," in *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, Kyoto, Japan, Oct. 2010, pp. 92–102.

[8] S. Lakshmanan, S. Sanadhya, and R. Sivakumar, "On link rate adaptation in 802.11n WLANs," in *Proceedings of IEEE INFOCOM 2011*, Shanghai, China, Apr. 2011, pp. 366–370.

[9] X. Chen, D. Qiao, J. Yu, and S. Choi, "Probabilistic-based rate adaptation for IEEE 802.11 WLANs," in *Proceedings of IEEE GLOBECOM 2007*, Washington, DC, Nov. 2007, pp. 4904–4908.

[10] J. Choi, J. Na, Y. Lim, K. Park, and C. Kim, "Collision-aware design of rate adaptation for multi-rate 802.11 WLANs," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 8, pp. 1366–1375, 2008.

[11] D. Nguyen and J. Garcia-Luna-Aceves, "A practical approach to rate adaptation for multi-antenna systems," in *Network Protocols (ICNP), 2011 19th IEEE International Conference on*, Vancouver, BC Canada, Oct. 2011, pp. 331–340.

[12] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *Proceedings of the Seventh Annual ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, July 2001, pp. 236–251.

[13] C. Chen, "Learning-based interference mitigation for wireless networks," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, May 2009. [Online]. Available: http://www.crhc.illinois.edu/wireless/papers/chen-phdthesis.pdf

[14] H. Jafarkhani, *Space-Time Coding Theory and Practice*. New York: Cambridge University Press, 2005.

[15] M. Chiani, M. Win, and A. Zanella, "On the capacity of spatially correlated MIMO rayleigh-fading channels," *Information Theory, IEEE Transactions on*, vol. 49, no. 10, pp. 2363–2371, 2003.

[16] E. Sengul, E. Akay, and E. Ayanoglu, "Diversity analysis of single and multiple beamforming," *Communications, IEEE Transactions on*, vol. 54, no. 6, pp. 990–993, 2006.

[17] G. Lebrun, J. Gao, and M. Faulkner, "MIMO transmission over a time-varying channel using SVD," *Wireless Communications, IEEE Transactions on*, vol. 4, no. 2, pp. 757–764, 2005.

[18] M. Larsen and A. Swindlehurst, "MIMO SVD-based multiplexing with imperfect channel knowledge," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, Dallas, Texas, Mar. 2010, pp. 3454–3457.

[19] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. New York: Cambridge University Press, 2005.

[20] K. Fall and K. Varadhan, *The ns Manual*, VINT, Nov. 2011, http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf.