CREATING SUBCHANNELS TO IMPROVE THROUGHPUT IN WIRELESS COMMUNICATION

BY

BRIAN B. PROULX

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, 2010

Urbana, Illinois

Adviser:

Professor Nitin H. Vaidya

ABSTRACT

This thesis focuses on the impact of splitting a single communciation channel into two separate subchannels. One subchannel is used to send large packets, and the other sends small packets. This split allows for the spectrum to be used more efficiently. Large throughput gains are possible, even with different mixes of traffic patterns.

The effect of different guard bands was studied using the USRP and GNU Radio. Two daughterboards were used for each USRP so that two channels were used for transmission. The two transmissions had the same throughput whether the channels were placed very far aparta or within a subcarrier of each other.

An implementation of this scheme was simulated using the NS-2 network simulator. The implications of splitting the channel are studied, showing the impact of different levels of small packet traffic on the overall network throughput. In these simulations, a simple topology was used. This topology was a base station with a varying, though small, number of surrounding nodes. Also studied was a simple bandwidth allocation algorithm that was generated from the results of the simulations.

To my parents, Martha and Tony, and my wife Kelly, for their love and support

ACKNOWLEDGMENTS

I would like to thank both Vijay Raman and Dr. Nitin Vaidya for all their help with this project. It certainly would not have been possible without their help.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

SNR	Signal to Noise Ratio
MAC	Media Access Control
ACK	Acknowledgement
AODV	Ad-hoc On-demand Distance Vector
QAM	Quadrate Amplitude Modulation
SIFS	Short Inter Frame Spacing
PLCP	Physical Layer Convergence Protocol
RTS	Request To Send
CTS	Clear To Send
FTP	File Transfer Protocol
TCP	Transmission Control Protocol
CBR	Constant Bit Rate
Mbps	Megabits per second
USRP	Universal Software Radio Peripheral

CHAPTER 1

INTRODUCTION

1.1 Multiple Interfaces

Multiple interfaces on a wireless node allow for a greater flexibility. The ability to utilize different spectrum widths allows the nodes to adapt to a changing traffic pattern. This thesis deals with converting one channel in the 802.11a spectrum into two subchannels. This split is based on the size of the packet. Small packets and long packets are separated by size, with long packets using one part of the channel and the small packets using the other.

This split allows the nodes to adjust the bandwidth based on the traffic flow. This thesis deals with an access point topology, where there are nodes that are all sending to a single node.

This split allows for greater flexiblity, but there is a tradeoff. A guard band is necessary to ensure the channels do not interfere with each other. Also, the allocation of the channels must be done correctly to ensure the throughput is increased.

Using two subchannels, as opposed to a single channel, can have significant benefits. For each packet to be sent, a node must contend for the channel, which requires some time. This time is considered wasted in terms of data transmission, since no data is sent. By splitting a single channel into two subchannels, the overhead of channel competition can be reduced since another channel is available for sending packets. Also, small packets require a short time to be sent, which means that the time overhead used in securing the channel is large compared to the time required to send a packet. This results in a poor spectrum efficiency. Spectrum effeciency can be improved by giving smaller packets a separate subchannel, with a lower rate. This low rate means that the packet requires more time to be sent across the channel, thus improving spectrum efficiency.

Contained within this framework is the relation between transmission rate and bandwidth. Bandwidth is a measure of frequency available for a transmitter to use, and is measured in Hertz. Rate is the amount of data that can be sent per unit time, and is measured in bits per second. These two quantities are directly related. A larger bandwidth results in a higher rate, and smaller bandwidth results in a lower rate. The calculations are explained in Section 4.4.

This work deals with simulations in order to find the optimal split between these two channels. The NS-2 software package allowed for a channel allocation algorithm to be developed. The metric used to evaluate this algorithm is the sum throughput of all the nodes.

CHAPTER 2

RELATED WORK

Primary work in this field was done by Chandra et al. [1], who present measurements using the Atheros chipset to draw conclusions about adjusting channel width. Their work focuses on a single point-to-point link, though the implications of its work for networking are very clear. In their paper, the primary result is that reducing the bandwidth of the channel both increases range and reduces the power used to send a signal. The increased range is due to the fact that the signal is now less susceptible to noise; it can still be decoded at lower SNR. This gain is due to the narrower bandwidths encountering less noise by only being a fraction of the original 20 MHz channel. The downside to this in networking is that the interference now carries farther than before. Their work was done using only four different channel widths on an Atheros WiFi card. Changing the bandwidth on the Atheros card also changed the timing, due to the fact that the reference timer is shared between the RF transceiver and the baseband processor. So changing the channel width changed the 802.11 timing intervals, as well as the OFDM symbols. It is unclear how much this change in the timing intervals affected the results. The other important takeaways from their paper are that adapting the channel width increased throughput, allowed for better fairness, and improved network throughput.

Also presented in [1] is SampleWidth, a communication protocol designed to find both the optimal rate and bandwidth. The bandwidth was divided discretely, in channels of 5, 10, 20, or 40 MHz. Their research was more focused on quickly finding the best rate to communicate at, as opposed to finding the best bandwidth. With only four choices, the correct bandwidth at which to operate can be quickly determined. SampleWidth uses a beacon interval of one second. During this second, the rate and the throughput were measured. If the rate was less than some constant, the channel was narrowed; if the rate was greater, a channel width was selected based on the throughput

measured.

Another related piece of work is a paper that proposed variable channel widths [2], which presents a protocol called variable width channels (VWID). This protocol uses a few point-to-point links and three options for channel bandwidths: 5, 10, or 20 MHz. It also takes into account the spacing for the channels in a 20 MHz bandwidth. This paper showed a large gain in throughput, as opposed to the standard 802.11 MAC and PHY layers.

The idea presented in [2] focuses on maximizing throughput by dividing the channel based on received signal strength at the access point. However, the traffic model is a constant bitrate source. This paper focuses on dynamic traffic, and thus a dynamically adaptive algorithm is required. However, [2] can be viewed as generating some relatively basic results.

Another paper that is related to this project is more focused on the routing implications of bandwidth and channel allocations [3]. The physical setup of the network used in that work is different from ours as well. Each radio has one antenna that is fixed at a center frequency. The other antenna on the radio is tunable to communicate to other nodes. The channel allocation is done dynamically at each hop. The algorithm uses the routing table to help make decisions based on flows to allocate different bandwidths and center frequencies. It relies on messages passed between neighbors. The network then adds a flow if it can be supported by all of the nodes within the network. The focus is more on quality of service (QoS) as opposed to maximizing throughput. This paper focuses more on the routing implications of variable bandwidth networking.

Another paper whose research relates to this thesis is [4], where a strong argument is put forth for using channelization in networking, as well as varying the bandwidth. The algorithm presented is also dynamic and can adjust to changing traffic rates. The paper also utilizes an extended reservation protocol to send multiple packets when the channel has been acquired after contention. The extended reservation protocol reduces the amount of overhead required to transmit data. Using multiple channels allows this extended reservation protocol to maintain fairness as well.

CHAPTER 3

GNU RADIO

3.1 Introduction

Another aspect of this thesis involved the universal software radio peripheral (USRP) and the GNU Radio. The USRP is a physical device developed by Ettus Research [5], that connects to a computer using a USB connection. This thesis used a USRP 1, also called the USRP Classic, and it will be referred to as simply USRP for the rest of the paper.

The USRP is designed to be used in conjunction with software radio packages, especially GNU Radio. The USRP is designed to operate on a frequency band, which is then converted down to baseband and then sampled. These samples are then sent over the USB connection to the computer, where the GNU Radio processes the samples. The frequencies available to the USRP depend on which daughterboard is connected to the motherboard. This thesis used the XCVR 2400 daughterboard, which has a frequency range from 2.4 to 2.5 GHz, and from 4.9 to 5.85 GHz. The exact center frequency of the USRP can be set via software commands. The USRP's bandwidth is determined by several different factors. The limit to the bandwidth is the 32 Mbps data rate supported by USB 2.0. To set this bandwidth, the decimation must be set on the USRP. Decimation is the amount the signal is reduced in sampling the bandwidth. The decimation must be set between 8 and 256, inclusive, which selects the bandwidth of the signal. The bandwidth of the signal is calculated by

$$\frac{64 \text{ MHz}}{d} \tag{3.1}$$

where d is the decimation of the USRP.

The samples from the USRP are then transferred over the USB connection to the computer, which passes them to the GNU Radio software. The GNU

Radio software is available at [6]. The GNU Radio software is designed in two levels. The top level is written in Python and is used only for connecting various processing blocks. This connection of processing blocks is called the flow graph. The processing blocks comprise the lower level and are written in C++. The GNU Radio software contains many various processing blocks that can be connected together to become a fully functional transceiver.

GNU Radio comes with some sample code. One of these samples is a fully functional OFDM transmitter and receiver. We changed this sample code to allow for split bandwidth transmission. Two daughterboards for each USRP allow for two channels to be used. One USRP was always transmitting and the other was always receiving. The two USRPs were placed approximately five feet apart. The difference in frequency was changed to allow for different guard bands.

The throughput was measured where each channel had backlogged traffic. The different guard bands were measured in intervals of a subcarrier. The two separate transmissions had the same throughput when separated 100 MHz apart and when separated only a subcarrier. This result is also supported by the NS-2 simulations.

CHAPTER 4

NS-2 IMPLEMENTATION

4.1 NS-2

NS-2 is an open source software package that is used to model network behavior. For this thesis, the wireless simulation capacity of NS-2 was used. Since NS-2 is open source, the source code is available for free and can be then modified to suit any need.

NS-2 models the network stack of a wireless node. It uses multiple layers to simulate the packet of information travelling up and down the stack. These different layers are separated to allow each to have its own model.

The topmost layer is the application layer, which is used to monitor the transmission and reception of packets. A packet is not received until it has reached the application layer, and it is considered in transmission once it has left the application layer. In NS-2, the application layer is used to keep track of which packets have been received and when. This allows for measuring throughput or delay.

The next layer is the routing layer and is responsible for collecting and maintaining routes from one node to another node. Packets that pass through this layer are stamped with information that allows other nodes to receive or forward the packet. This layer will also generate routes from one node to another should a packet be sent that does not have a route setup yet. NS-2 fully models the routing layer. Each node maintains its routing table and is responsible for route discovery and maintenance.

Below the routing layer is the link layer. The link layer is concerned with local data transmission. It defines a data frame. It is responsible for MAC addressing. In NS-2, the link layer adds the MAC header and determines which MAC address to send the packet to.

The next layer in the stack is the MAC layer, which determines the method

for accessing the wireless medium. Avoiding collisions and determining when the channel is clear are the primary functions of this layer. In NS-2 most of the work done traditionally in the physical layer is done here. Determining whether a packet is able to be received on the basis of its energy is performed here. Also performed is backoff and acknowledgement of received data packets.

Beneath the MAC layer is the physical layer, which is for actually sending electromagnetic signals over the air. It is also responsible for demodulation of received packets. NS-2 uses this layer only for energy dispersion and the wireless model for energy propagation. No actual modulation or demodulation is done in NS-2.

For further details about the 802.11 MAC, see the note written by Liu [7]. It covers basic sending and receiving, as well as outlines the functions and variables contained in each layer.

4.2 Overview

Originally, NS-2 implements only a single wireless interface per node. For this thesis, each node has two interfaces. This alteration required changing many aspects of the NS-2 code, in order to allow for each node to have more than one interface. Since NS-2 is written in both C++ and Tcl, there are changes that need to be made to each aspect of the program. Most of the changes made to NS-2 for this thesis are outlined in [8] and are presented here for completeness.

One key aspect to this conversion is that multiple channels are handled at the Tcl level. This allows the C++ code to be much cleaner, and allows greater flexibility at the simulation level. Each change to the channel parameters does not require the source to be rebuilt. The decision about which interface to use is made at the routing layer. For this thesis, the two interfaces are split based on the size of the packet. This split threshold is set at run time, through the Tcl script.

4.3 Tcl Code

There are many changes made at the Tcl layer in this thesis. They are necessary to set up the MobileNode correctly. Because MobileNode has multiple interfaces, more data needs to be given to it at the outset of the program.

In ns-lib.tcl, the procedure change-numifs is used to create a new variable for MobileNode called numifs_. This procedure also sets the variable appropriately. This variable represents the number of interfaces available at each node. Another change to MobileNode is the related function get-numifs. This function returns the number of interfaces for each node, and for backwards compatibility, it returns an empty string should the node only have a single interface.

A function for setting a channel for each interface is used as well, called add-channel. This is important because it requires the channel to be set up before calling node-config, which is changed to allow for multiple interfaces as well. This function now adds a channel array to keep track of each channel for the interfaces. In create-wireless-node, code is added to initialize each interface with its own packet queue, MAC layer, and physical layer.

The file ns-mobilenode.tcl has changes made to it as well for this thesis. For each target added, all of the interfaces must be hooked up properly. Also, each interface must keep its own ARP table, so that packets are sent to the correct interface of the recipient node.

The changes made for this project allow the simulation to be changed without rebuilding the source code, which allows for a large amount of flexibility when running simulations. The Tcl scripts handle the basics such as configuring the node properly and maintaining the correct number of interfaces per channel. The rest of the changes for this project are made in the C++ portion of NS-2.

4.4 C++ Code

The definition of MobileNode needs to be changed in the C++ code, which requires editing both mobilenode.h and mobilenode.cc. MobileNode must now keep track of the channels it is a member of. Previously, this was simple, but now that the node is attached to multiple channels, the next

and previous nodes now need to be a list, so that each interface knows the next and previous nodes on that particular channel. The next and previous nodes are used to ensure that a packet sent on a channel is delivered to each MobileNode. The getLoc function must be updated so that is correctly returns the position of the node.

The file channel.cc has the code to determine which MobileNodes can receive the packet. Since this functionality depends on both next and previous, as defined in MobileNode, these arrays must be indexed at the correct location to get the next and previous nodes attached to the current channel. Also, when scheduling a delivery to a node, the correct interface must receive the packet.

The mac802_11.cc file needs to have a single line added to it. In the recv function, the packet's header is stamped with the index of the MAC, which allows the higher layers to know which interface received that packet.

The most important aspect of NS-2 to change is the routing layer. This thesis uses AODV as the routing protocol, so aodv.h and aodv.cc are the files that need changing. The routing layer is where the decision to send a packet to a specific interfaced is made. This decision requires each interface to have its own link layer and corresponding queue. Thus, the number of interfaces and a list of each link layer and queue is maintained. The command function of AODV is changed so that these new variables can be updated by the Tcl script, which allows for different simulations to use different numbers of interfaces without changing the underlying C++ code.

For this thesis, there are two interfaces. This means that there are two routing tables, one for each interface. Each routing table has identical entries, since whenever an entry is made into one table, the corresponding table can make the same entry. This is true due to the assumption that each node has two interfaces, and that all of the interfaces have the same bandwidth. Broadcast packets are always sent to both interfaces.

Another change to the C++ code is the RATE function used at the MAC layer. This change is new to the implementation for this thesis and is not contained in [8]. This function determines at what rate to send the packet based on the amount of bandwidth available. Some packets, such as ARP packets, are sent out at a default 6 Mbps. Other packets are sent based on the bandwidth allocated to that interface. All rates are chosen based on a set of assumptions. These are that a modulation of 64 QAM is used, with a coding

rate of 3/4. The full 20 MHz bandwidth is divided into 48 subcarriers, and with a bandwidth of 20 MHz; the symbol time is 4 μs. The symbol time is inversely proportional to the percentage of bandwidth. Each subchannel has 48 subcarriers. Some subcarriers are subtracted from the larger bandwidth if a guard band is used, which can be specified in the Tcl script.

The rate function is designed as follows. The number of subcarriers is always fixed at 48 per subchannel. The symbol time is changed by the percentage of bandwidth per channel, as shown in Equation 4.1. The number of subcarriers in the larger bandwidth channel is reduced by the number of subchannels needed for the guard band. The change in symbol time is responsible for the increase or decline in rate.

$$T_{symbol} = \frac{4\ \mu\text{s}}{\%_{bandwidth}} \tag{4.1}$$

CHAPTER 5

RESULTS

5.1 Overview

The double interface network model developed for this thesis project was tested using NS-2. Testing the model is the main focus of this thesis. The NS-2 simulations show the impact of splitting a single channel into two sub-channels. The metric for evaulation used was throughput. Delay was not measured at this time, but would be a source of future testing. The throughput of the two interface model was compared against a copy of NS-2 that was not modified in any way. The Tcl script used to run the simulation was slightly altered to fit the unaltered copy of NS-2, but the changes are more cosmetic than substantial. In order to obtain better data, each simulation was run five times. Each run used a different seed to generate the random numbers, which are used throughout NS-2. Many simulation parameters were unchanged between runs. These are listed in the next two paragraphs.

The routing protocol used was AODV, since that was the protocol that had been changed to use the multiple interfaces. The channel type used was WirelessChannel, and the radio propagation model was Shadowing, using the OmniAntenna model for the antennae. The IFQ was always a PriQueue, which is a priority queue, and its length was 50 packets.

802.11a parameters were used to set the MAC and physical layer parameters. The minimum contention window was 15, and the maximum was 1023 slots. The slot time was 9 μs and the SIFS length was 16 μs. The PLCP length was 24 and the data rate for it was 6 Mbps. The preamble length was set to 96 bytes. The RTS/CTS handshake was not used.

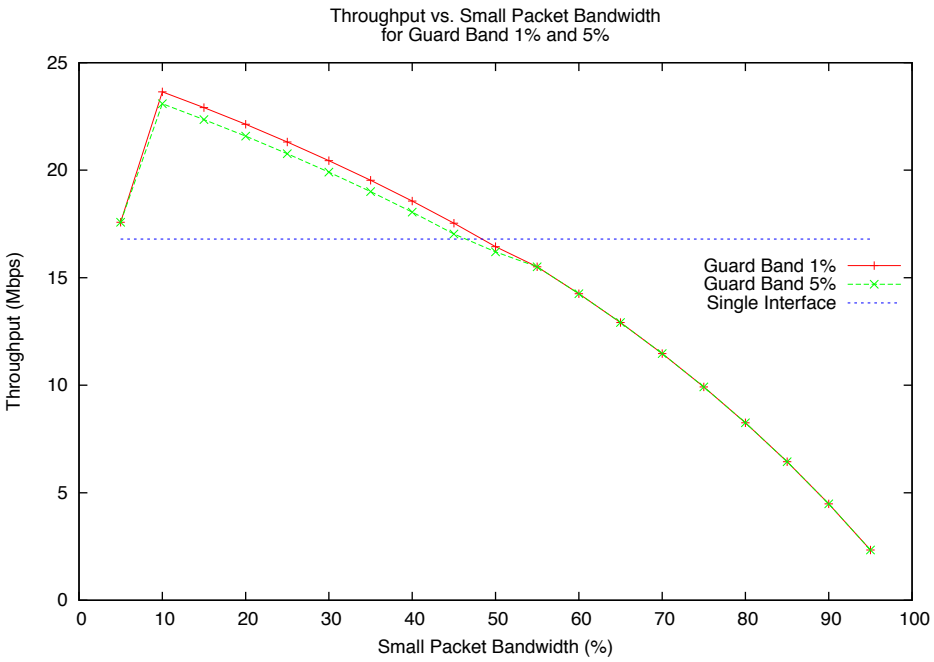

Figure 5.1: Throughput of Two Nodes Running TCP

5.2 Two Node Results

The first test of the new network stack was done using two nodes. These two nodes each had two interfaces that were split by packet size. One interface was set to send and receive packets that are less than or equal to 128 bytes. The other channel was for packets larger than 128 bytes. The simulation was run for 100 seconds. The two nodes were placed 10 meters apart. One node used the FTP application to send data over TCP to the other node. These data packets were 1000 bytes long. There were two variables in these simulations: the guard band percentage and the percentage of bandwidth given to the channel carrying the smaller packets. All of the data flows from one node to another.

The results are shown in Figure 5.1. The two interface model outperforms the single interface model until the percentage of bandwidth allocated to the small packets reaches about 50%. The small packets need some bandwidth to quickly send the ACK packets so that TCP can send more data packets. The percentage change is shown in Figure 5.2, which is the percentage change compared with the baseline, single interface case. The best percentage improvement occurs when the small packets have a bandwidth of 10%. Also,

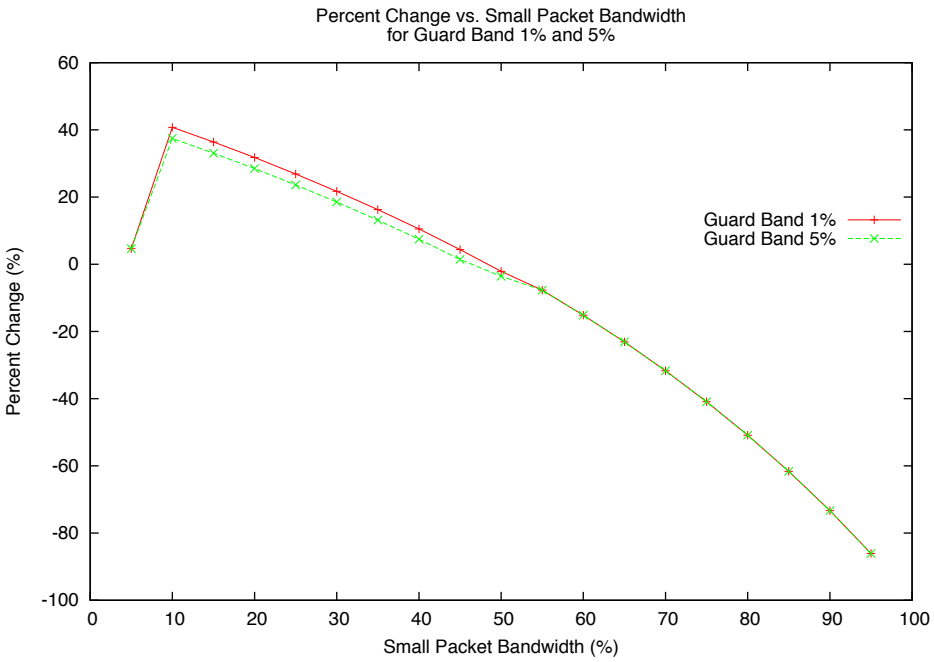

Figure 5.2: Percentage Change of Two Nodes Running TCP

Figure 5.2 shows that there is almost no difference between the 1% and 5% guard bands, which is due to the fact that this small bandwidth difference is outweighed by the channel saturating. That is, both guard bands allow for transmission rates above what the channel can support using TCP.

The next simulation involved varying only the percentage of the bandwidth given to the small packets, because the previous simulation showed very little degree of difference for the differences in the guard band. So the guard band is fixed at 5% and will remain that way for the rest of the simulations. This time, UDP will be used to transmit packets, instead of TCP. In order to generate both small and large packets, two CBR flows were set up between the nodes. One flow was set to send packets of 1000 bytes and the other flow was set to send packets of 100 bytes. Both flows were set to send at 20 Mbps.

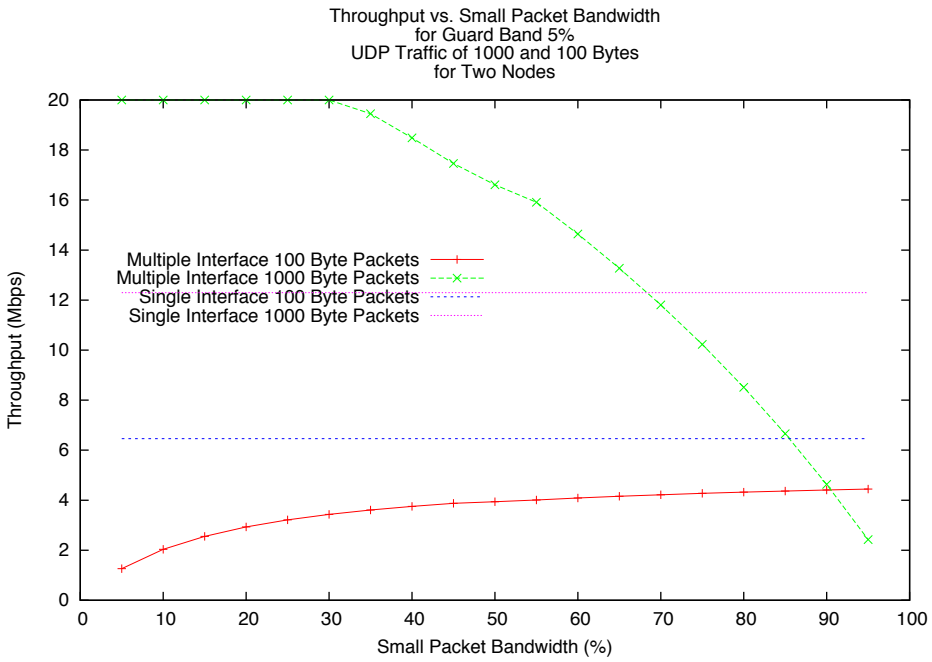

Figure 5.3: Two Nodes Running UDP

The results, which are shown in Figure 5.3, are encouraging. The two interface model outperforms the standard model until about 60% bandwidth for the small packet channel. The smaller packet flow does significantly better, and the larger packet flow improves as well. The sum throughput can be seen in Figure 5.4. This behavior can be explained by the reduction in overhead by splitting the channel. The small UDP packets are generated more frequently, thus introducing more overhead. Since the larger packets do not have to contend with the smaller packets, they are sent more often. When the bandwidth given to the small packets is too large, fewer 1000 byte packets can be sent. These 1000 byte packets have a higher spectrum efficiency, so reducing the number of them reduces the overall throughput.

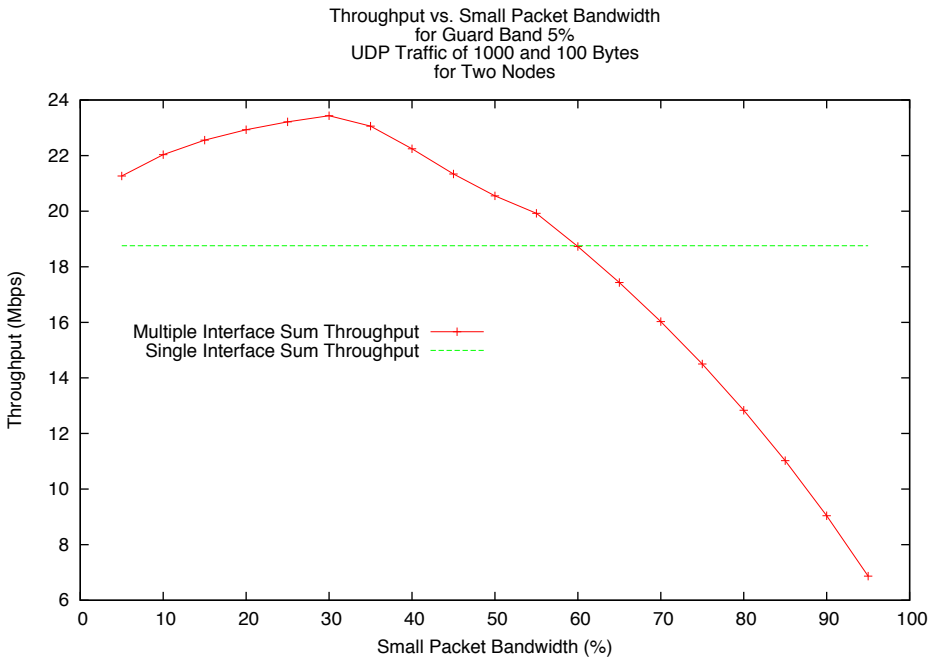

Figure 5.4: Sum Throughput of Two Nodes Running UDP

The next test of the double interface involved a different ratio of small to large packets. All other variables were kept the same as for the previous test. The new mix was four 1000 byte packets to every one 100 byte packet. The 1000 byte packets were sent at 20 Mbps, the same as before, but the smaller packets, 100 bytes, were sent at a rate of 5 Mbps.

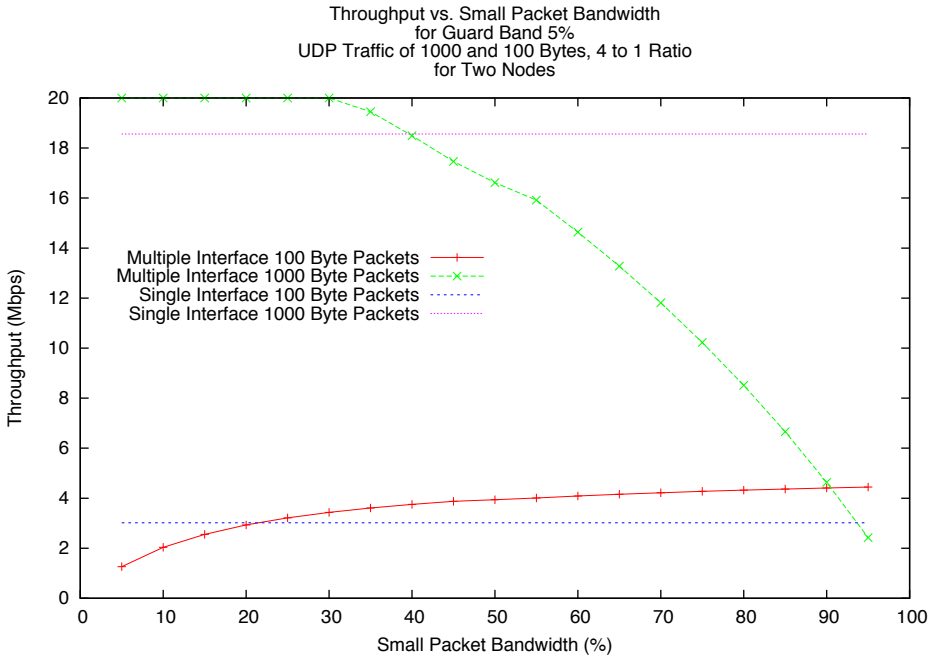

Figure 5.5: Throughputs of Two Nodes Running UDP with a 4:1 Packet Ratio

The results of this simulation are shown in Figure 5.5. The single interface model performs better than the double interface model starting at about 45% for the small bandwidth percentage. This represents the area where the increase in throughput for the smaller packets is not enough to balance out the decrease in throughput of the larger packets.

The same parameters with a single change were used again. Instead of a 4:1 ratio, the ratio set was 20:1. The large packets were sent at a rate of 20 Mbps and the small packets were sent at a rate of 1 Mbps. The results are shown in Figures 5.6 and 5.7. The single interface is able to support the total throughput of 21 Mbps despite the MAC layer overhead. The double interface model can only match this best-case scenario; then the bandwidth of the larger interface becomes too small to support a 20 Mbps flow, and the performance degrades.

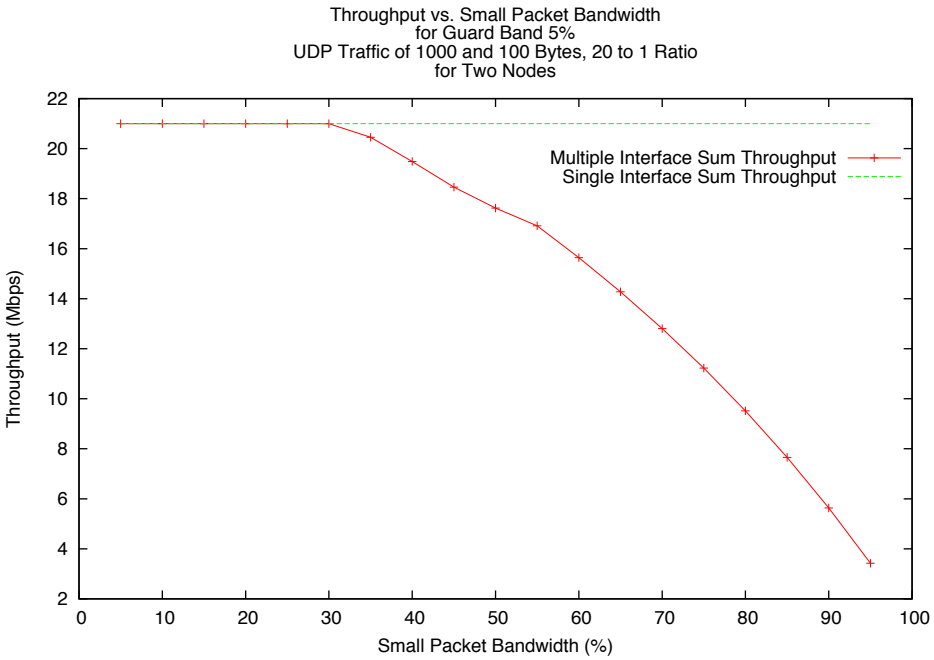

Figure 5.6: Sum Throughput of Two Nodes Running UDP with a 20:1 Packet Ratio

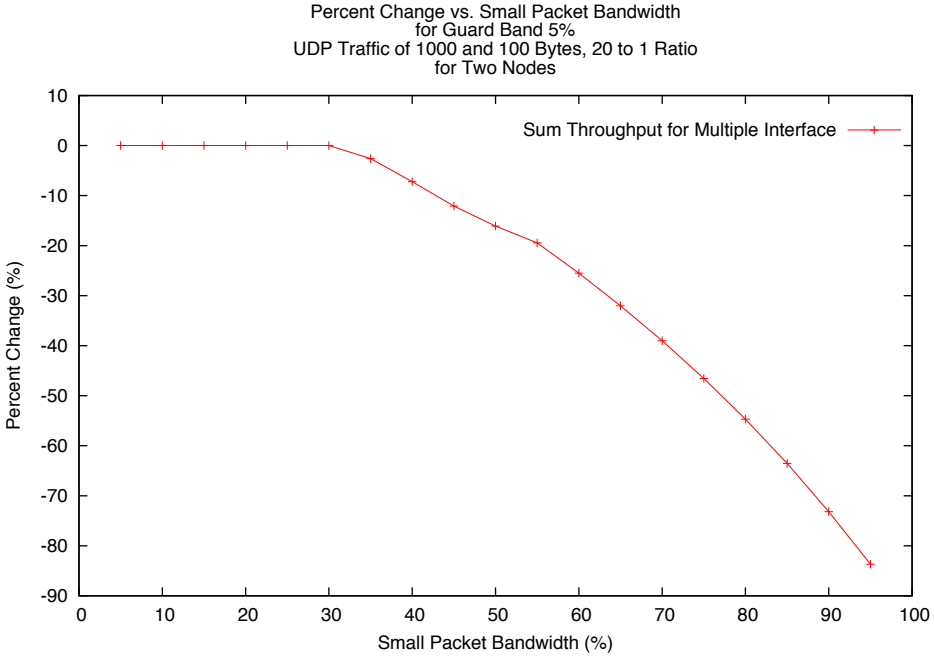

Figure 5.7: Percentage Change of Two Nodes at a 20:1 Packet Ratio

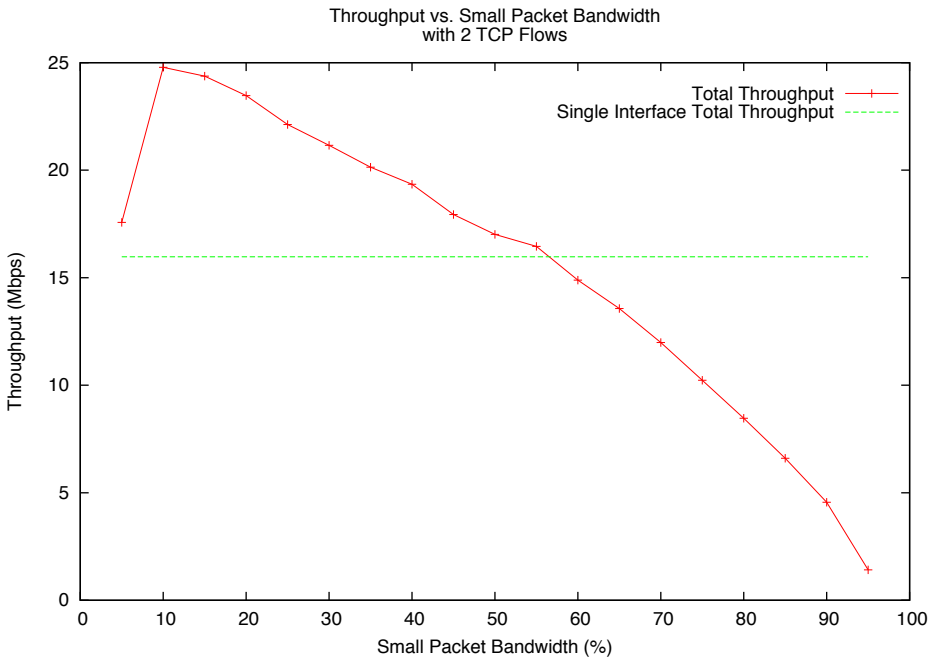

Figure 5.8: Throughput of Three Nodes Using TCP

5.3 Multiple Node Results

The next aspect of this thesis is evaluating the performance of the two interface model with multiple nodes. In these simulations, one access point acts as the sink for all of the traffic in the network. All other nodes are placed around the access point in a circle with a radius of ten meters. Different traffic patterns were used to simulate various types of network usage. The guard band remains a conservative 5% for all of these tests. All of the tests were run five times and the results averaged to generate these plots. The timescale for all simulations was 50 seconds.

The first test used three nodes, an access point, and two clients. The two clients were transmitting FTP data over TCP. Like the two node case before, the most improvement over the stanard case was when the percentage of the bandwidth allocated to the small packets was 10%. Again, when the small packet percentage becomes 60%, the two interface model does worse. These trends can be seen in Figure 5.8.

The next test again used FTP traffic over TCP, this time using four nodes in total. The results are very similar to the three node case. The results at the peak of 10% small packet bandwidth are slightly better than the three

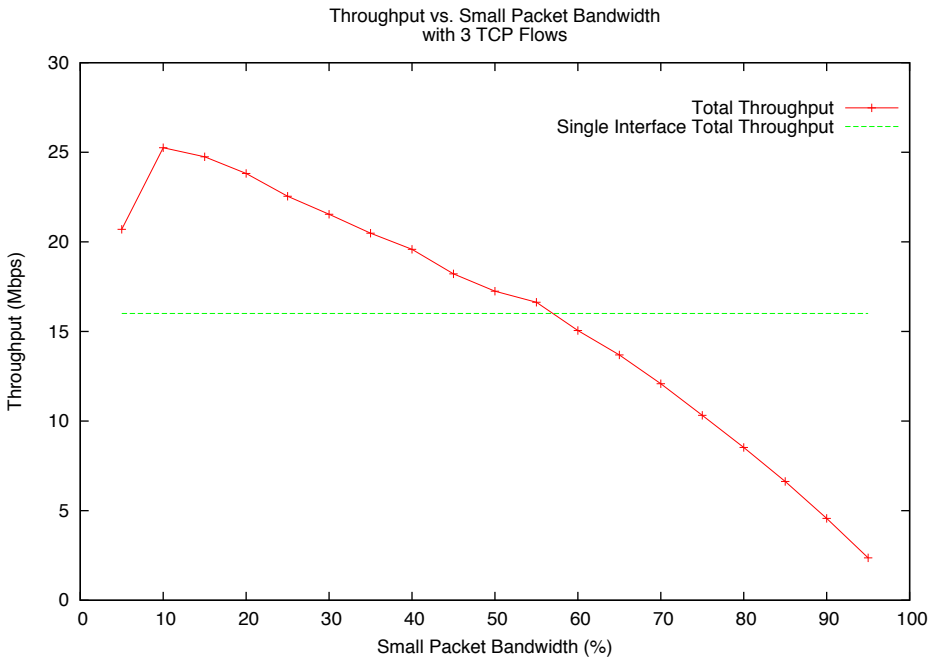

Figure 5.9: Throughput of Four Nodes Using TCP

node case. The basic shape of the graph remains the same, which can be seen in Figure 5.9.

The results in the three node and four node cases, Figures 5.8 and 5.9, respectively, are different from the results of the two node case, Figure 5.1. The three and four node cases both show a performance increase in the two interface model compared to the single interface model for a larger subset of small packet bandwidth. This behavior is due to the fact that with more nodes, more small packet bandwidth is needed since there are more ACK packets to be sent. TCP throughput depends on receiving these ACK packets quickly.

The test results with five nodes, four clients, and the access point for FTP traffic over TCP were generally similar to the previous tests, as can be seen in Figure 5.10. The noticeable difference is that the two interface model does worse than the single interface model starting at 50%. This behavior is slightly different from the fewer node cases. There is no longer a sharp increase from 5% to 10% small packet bandwidth, which shows the impact of increasing nodes. Having more nodes in the network requires a greater reliance on the small ACK packets being delivered to allow TCP throughput to increase.

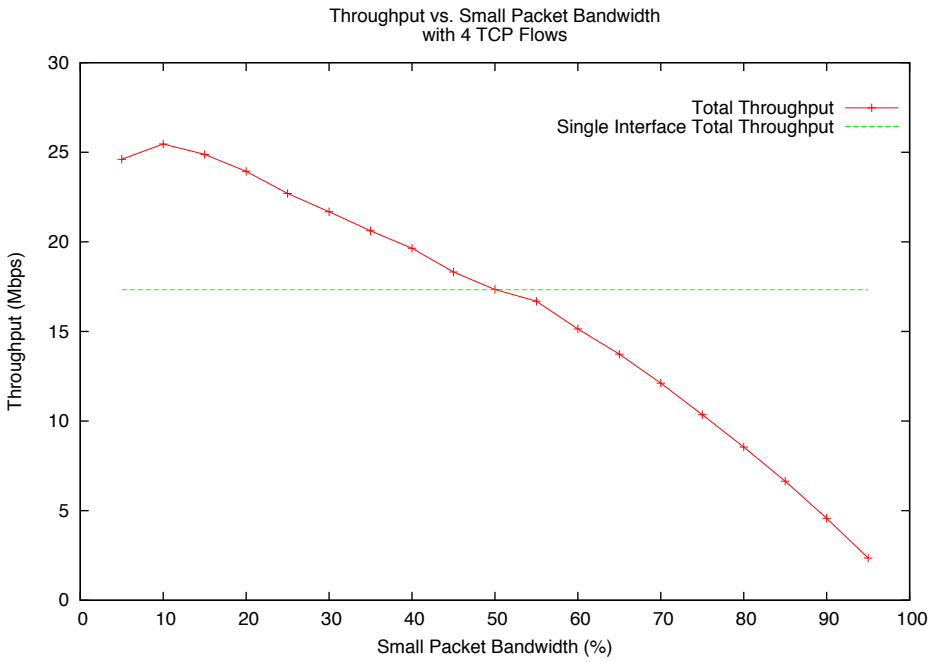

Figure 5.10: Throughput of Five Nodes Using TCP

5.4 Mixed Traffic Flows

The next step in evaulating the two interface model is to use different traffic mixes, which are a combination of TCP flows and UDP flows. The TCP flow is again set up using an FTP transfer so the channel is saturated. The UDP flows are flows at 1 Mbps of 100 byte packets. These 100 byte packets are always sent on the small packet channel due to their size. The all-TCP flow case is discussed in the previous section. The all-UDP case does not make sense because all the packets should be sent on the small packet channel and a throughput decrease would occur due to the overhead. The simulations were run for 50 seconds each, and there were five runs averaged together to get the results. All of the traffic flows started at the same time.

The first test involved three nodes, a base station, and two senders. The two senders were set to different traffic patterns. One was a TCP flow and the other was a 1 Mbps 100 byte cbr packet flow. The results are shown in Figure 5.11. The peak throughput occurs at 20% for the small packet bandwidth. The increased number of small packets due to the UDP flow requires more bandwidth. The two interface model also is better until the small bandwidth reaches 65%.

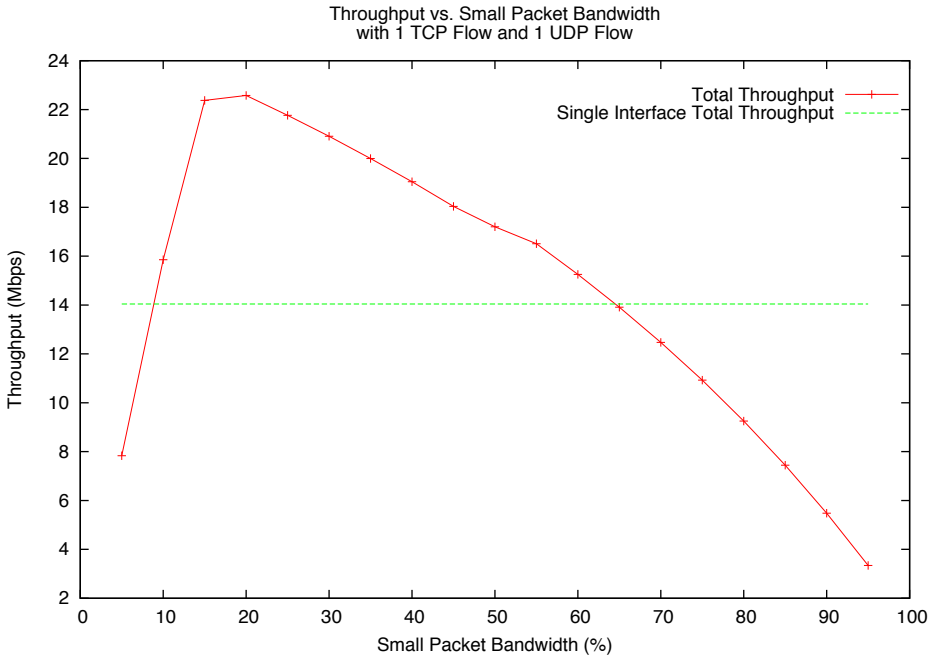

Figure 5.11: Throughput of Three Nodes With One TCP Flow and One UDP Flow

The next test involved four nodes. There were three sending nodes and a base station. There are two cases for this test. One section uses one TCP flow and two UDP flows, and the other uses two TCP flows and one UDP flow. The two TCP flow and one UDP flow cases are shown in Figure 5.12. The singular UDP flow causes the best throughput to be acheived at 20%. This allows the TCP data packets to be transmitted quickly.

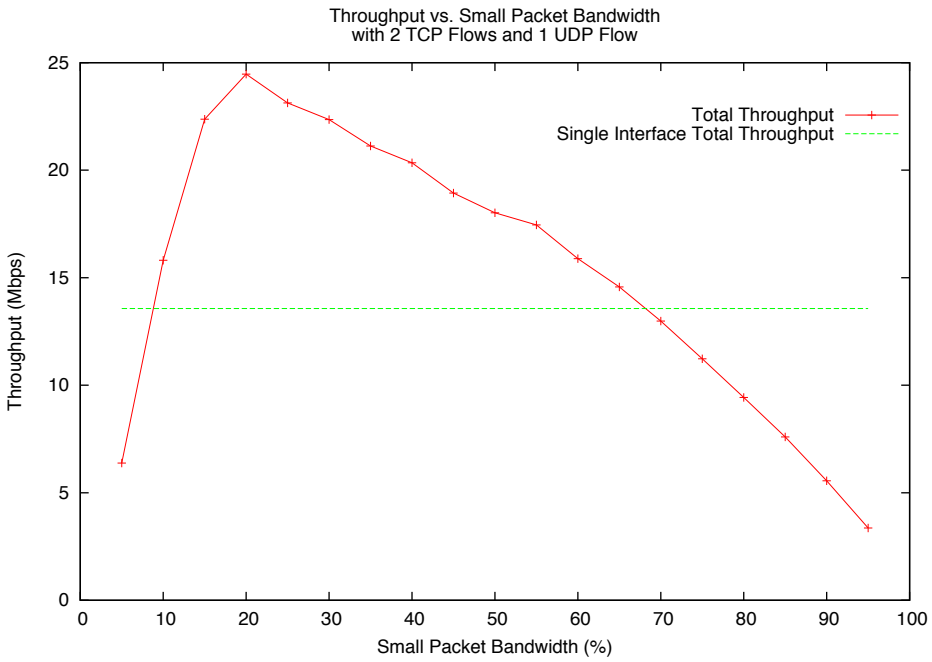

Figure 5.12: Throughput of Four Nodes with Two TCP Flows and One UDP Flow

The one TCP flow and two UDP flow cases are shown in Figure 5.13. The UDP flows cause the peak throughput to occur at 30%. Also, fewer large data packets are being sent. This peak in the graph is also sharper than previous peaks, as the balance between UDP throughput and TCP data packets is delicate.

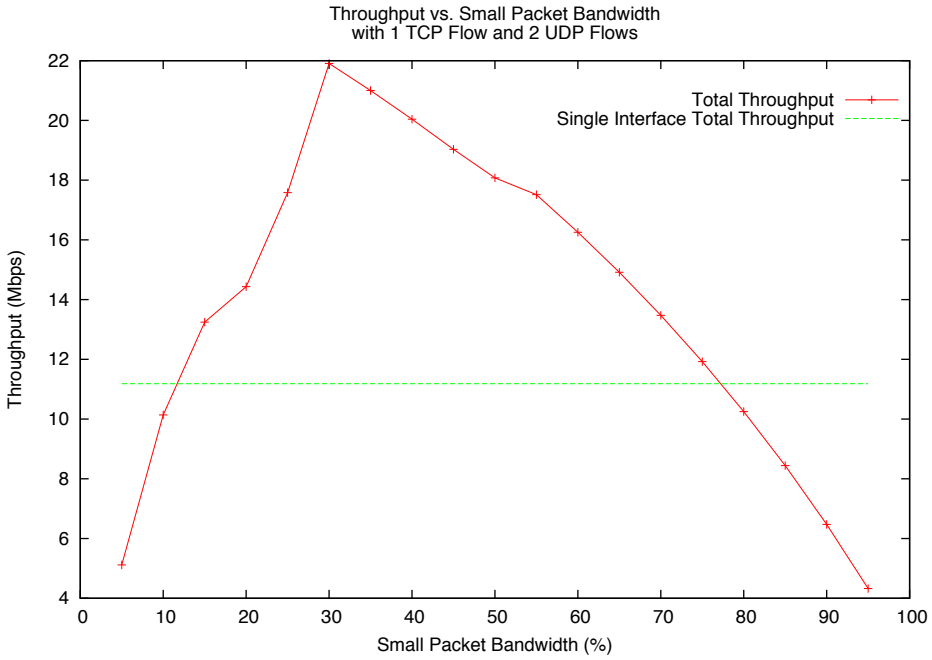

Figure 5.13: Throughput of Four Nodes with One TCP Flow and Two UDP Flows

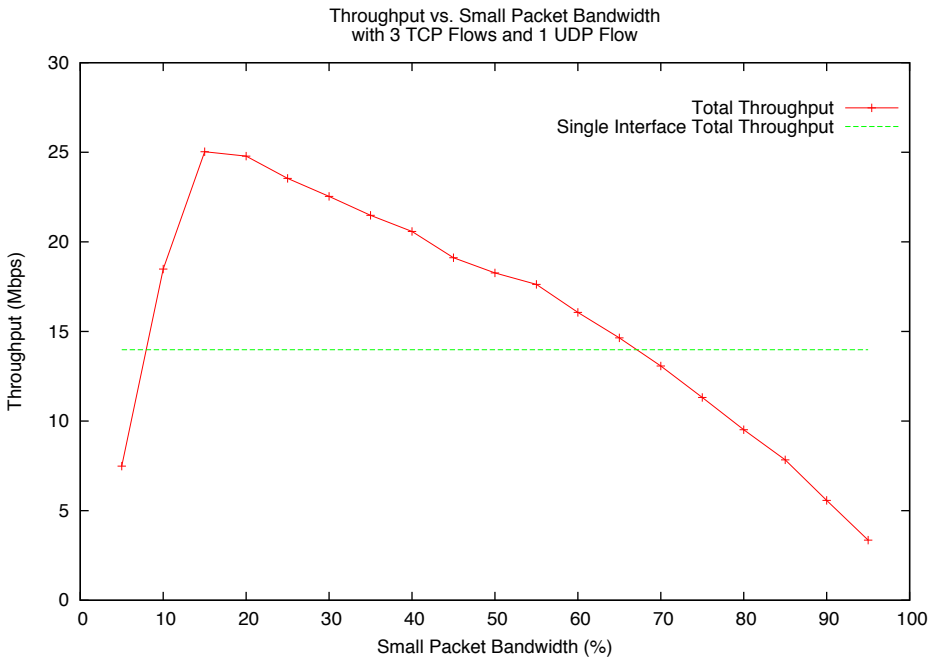

Figure 5.14: Throughput of Five Nodes with Three TCP Flows and One UDP Flow

The peak improvement has shifted somewhat from the two TCP case to the one TCP case. The UDP flows are comprised entirely of small packets; therefore, having more small packets in the network corresponds to having more UDP flows, which requires more small packet bandwidth to achieve high throughputs. Also, there is a large throughput gain over the single interface model since the spectrum is being used more efficiently. The small packets are sent at a rate where the transmission time of the packet is much larger than the contention overhead.

The five node test involved four sending nodes and one base station. The UDP flows are still at 1 Mbps. There are three cases, one with three TCP flows and one UDP flow, one with two TCP flows and two UDP flows, and one with one TCP flow and three UDP flows.

The results are shown in Figures 5.14, 5.15, and 5.16. Figure 5.14 shows the throughput of the three TCP flow and one UDP flow case. The throughput has a large gain where the TCP data packets have a large bandwidth, maxing out at 15%. The throughput then decreases as the TCP throughput drops rapidly while the UDP throughput climbs slowly.

The two TCP and two UDP flow case is shown in Figure 5.15. At 30%, the

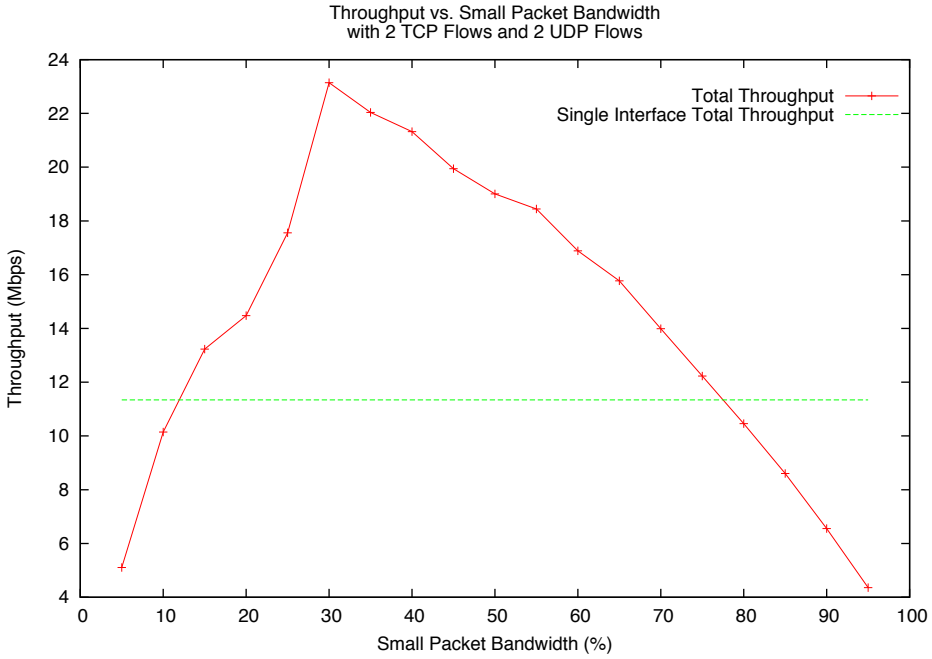

Figure 5.15: Throughput of Five Nodes with Two TCP Flows and Two UDP Flows

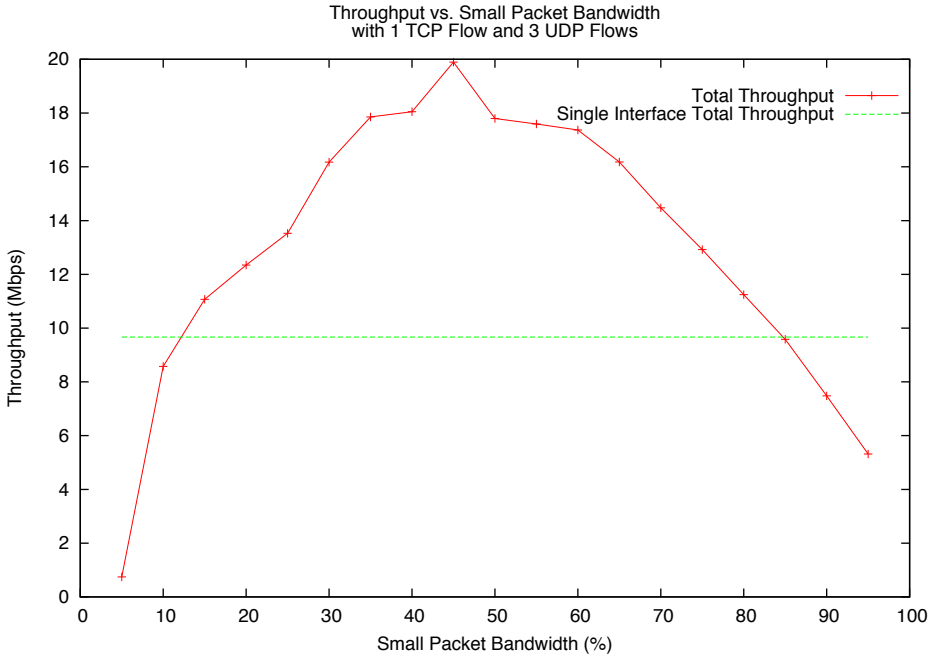

Figure 5.16: Throughput of Five Nodes with One TCP Flow and Three UDP Flows

throughput has reached a maximum improvement. The TCP data packets are still sent quickly, but the UDP data packets have enough bandwidth to be sent at a reasonable rate. The limitations of the single interface model become especially apparent in this figure. The single interface model causes the TCP throughput to be limited by channel contention, whereas the two interface model can avoid some contention.

The final result, with one TCP flow and three UDP flows, is shown in Figure 5.16. There is again a large throughput improvement due to the ability of the two interface model to allow the TCP data flow to achieve a high rate.

A comparison across the different five-node configurations shows some interesting dynamics. One dynamic is that the peak improvement occurs at high small bandwidth percentages with an increasing number of small packets. This improvement is to be expected as more small packets should require more bandwidth to be effectively transmitted. Also, splitting the channels allows for almost a doubling in throughput in the best case for each traffic configuration. This result shows the impact of utilizing the channel effectively to minimize the impact of overhead.

All of these measurements highlight a single effect. Having more channels reduces the amount of overhead by reducing the number of flows competing for the channel. Ideally, each flow could have its own subchannel; thus, there would be no contention overhead. This reduction of overhead is also related to the notion of spectrum efficiency. The best spectrum efficiency is to transmit data all the time. Any overhead relating to control packets, contention, or channel sensing reduces spectrum efficiency.

CHAPTER 6

VARYING TRAFFIC

6.1 Dynamic Traffic Mix

The next application of this multiple interface protocol is to use it in a small network. This small network has a single base station, which acts as the receiver for all of the traffic generated at the other nodes. These other nodes range in number from one to four. These transmitting nodes each choose their own traffic pattern from a set. These simulations were run with the same parameters as before for the physical and MAC layers. Also, each simulation was run for 50 seconds; and each configuration was run five times, to create an average for that configuration.

The result of all of the previous simulations is to provide a lookup table for a protocol that will select the channel bandwidths. This table allows the base station to select the optimal bandwidth allocated to the small packets. The results from the previous simulations are shown in Table 6.1.

Table 6.1: Small Bandwidth Table

Tcp Nodes	UDP Nodes	Small Bandwidth
1	0	10
0	1	95
2	0	10
1	1	20
0	2	95
3	0	10
2	1	20
1	2	30
0	3	95
4	0	10
3	1	15
2	2	30
1	3	45
0	4	95

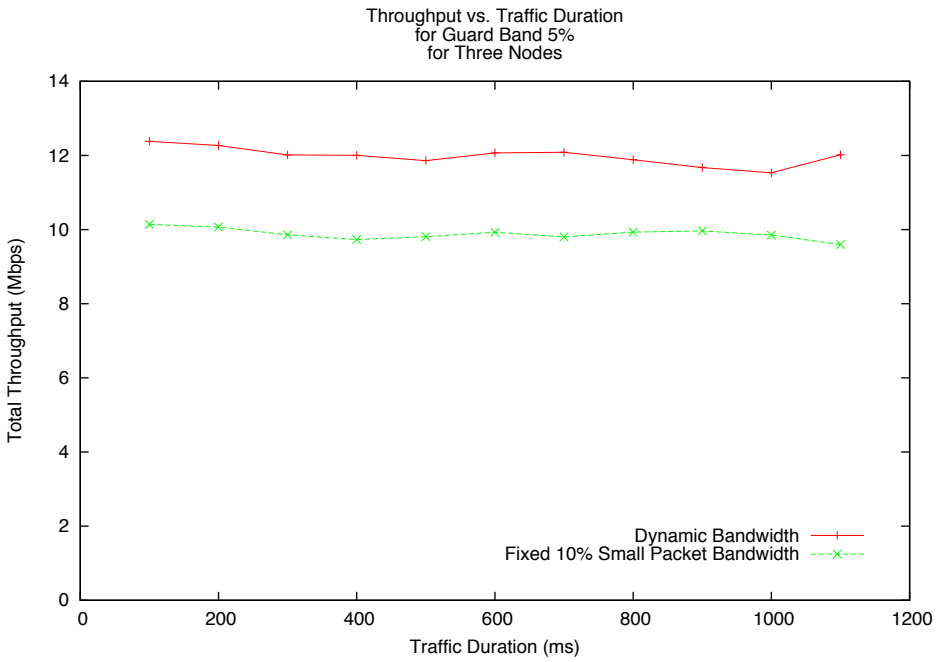

Figure 6.1: Three Nodes With Different Time Scales and Random Traffic

In order to test this dynamic scheme, different traffic patterns and time scales were used. A script was used to generate random traffic patterns. The script randomly chose from three different options in each time interval for each node. These options were no traffic, a TCP flow using FTP, and a 1 Mbps UDP flow with packet size of 100 bytes. The simulations ran on the premise that all of the nodes knew the precise traffic mix, so the subchannels were allocated using the lookup table (Table 6.1.) The timescale refers to the amount of time between picking a new traffic pattern. These timescales were varied from 100 ms to 1100 ms. There were 30 different traffic patterns generated for each timescale, and each traffic pattern was run five times for 50 seconds each.

To compare to a fixed scheme, these same 30 traffic patterns were run with the percentage of bandwidth given to small packets set to a fixed 10%. Again, each of these traffic patterns was run five times for 50 seconds each. The term "Traffic Duration" in Figures 6.1, 6.2, and 6.3 refers to the time between each node picking a new traffic pattern. difference between the two schemes.

The first case was three nodes, and the results are shown in Figure 6.1. From the graph, it can be seen that there is little difference in the throughput

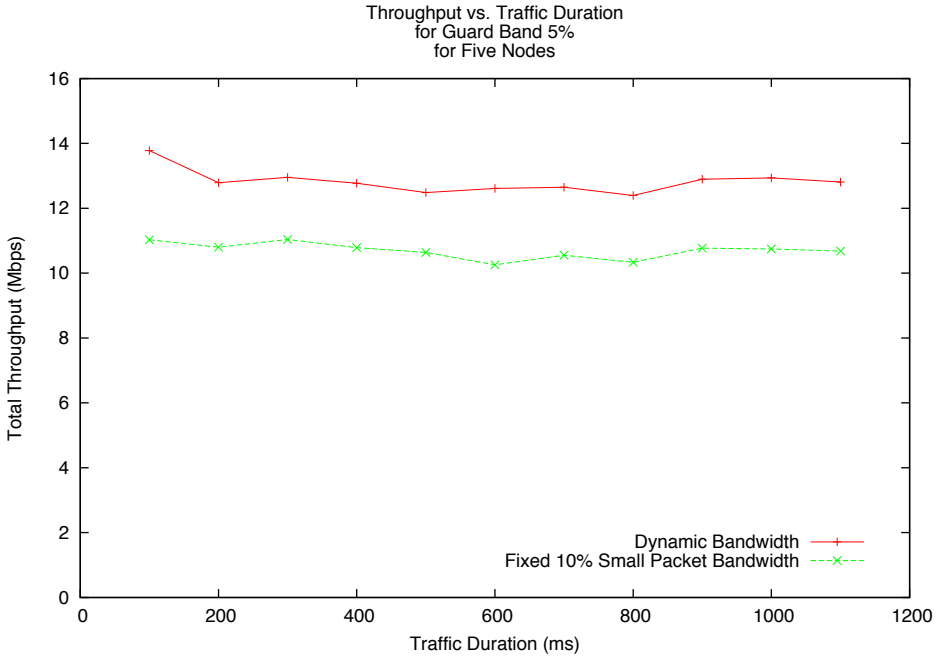

Figure 6.2: Four Nodes With Different Time Scales and Random Traffic

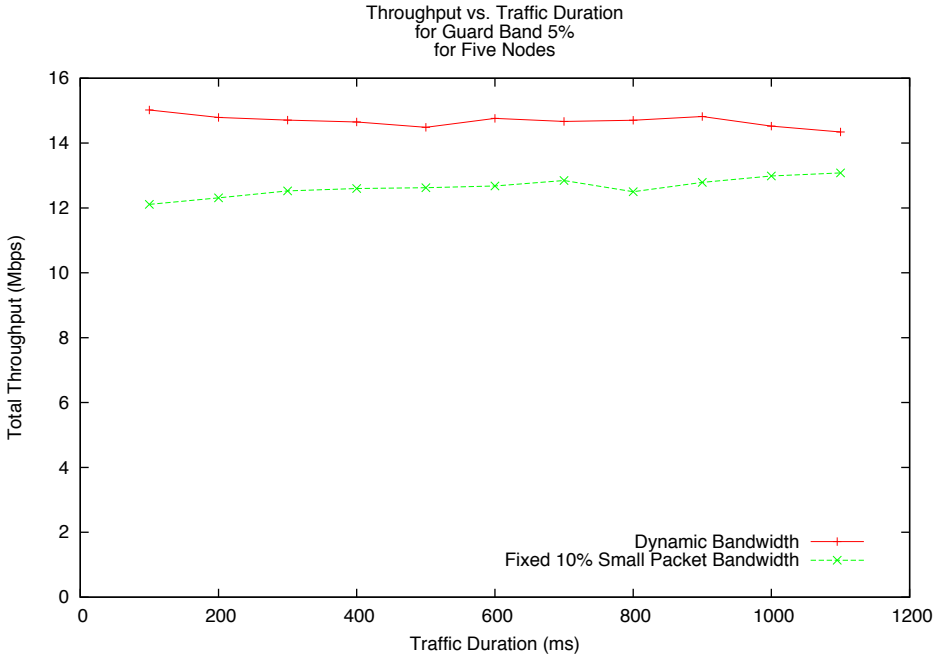

Figure 6.3: Five Nodes With Different Time Scales and Random Traffic

based on the time scale. For all time scales, the dynamic scheme outperformed the static scheme by approximately 2 Mbps.

The four node case is very similar to the three node case, as can be seen in Figure 6.2. The dynamic scheme again outperforms the fixed bandwidth scheme.

The effect of having more nodes is clearly seen in the results for five nodes in Figure 6.3. The dynamic scheme outperforms the static scheme by roughly 2.5 Mbps. There are now enough nodes in the network to show the

These results reveal a few key insights. The first is that the duration of the traffic has no effect on the improvement seen in the dynamic scheme. No matter the duration of the different traffic patterns, the result is that the dynamic case is always around 2 Mbps better. This is expected since the duration of the patterns should have little effect on the overall performance of the scheme.

Another insight is that this is the maximum improvement possible compared to the fixed scheme. This dynamic scheme has been optimized for the exact nature of the traffic. Also, each node, including the base station, has instant knowledge of the network and its traffic patterns. This knowledge is gained without any message passing and is always up to date. Any real scheme will not have these advantages and, thus, must be no better than this dynamic scheme.

CHAPTER 7

CONCLUSION

This thesis focused on splitting a single channel into two subchannels. One subchannel was dedicated to transmitting large packets, and the other was for small packets. The percentage of the bandwidth allocated to each subchannel was modified so that the impact could be studied. This splitting allows for throughput gains in the network. These throughput gains result from both a decrease in contention, and an increase in spectrum efficiency.

The USRP and the GNU Radio were used to study the effect of the guard band required between the two subchannels. The subchannels can have a minimal guard band and still have the throughput unaffected.

The NS-2 simulator was used to implement this protocol. Large throughput gains, some having as much as 100% gain, were possible with this splitting of bandwidth. This improvement occurs despite using some bandwidth as a guard band, reducing the overall spectrum to be used for data transmission. Both TCP and UDP flows were studied using this method. Combinations of these flows, even time varying combinations, show great improvement in throughput.

There are many directions for future work. One direction is to extend the bandwidth allocation algorithm. This implementation required an oracle to instantly spread network information to each node. A realistic bandwidth allocation algorithm would use messages, and perhaps even a dedicated control channel, to maximize throughput. Another direction would be to further study the fairness of this idea. Some fairness results were shown, but a more thorough treatment could be done. Finally, analytical results could certainly be generated, in the same vein as [9], which would quantify the different tradeoffs in this scheme.

REFERENCES

[1] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl, "A case for adapting channel width in wireless networks," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, 2008, pp. 135–146.

[2] R. Gummadi, R. Patra, H. Balakrishnan, and E. Brewer, "Interference avoidance and control," in *7th ACM Workshop on Hot Topics in Networks (Hotnets-VII)*, Calgary, Canada, October 2008, pp. 13–18.

[3] V. Raman and M. Caesar, "A practical approach for providing QoS in multichannel ad-hoc networks using spectrum width adaptation," in *IEEE Global Telecommunications Conference, 2009 (GLOBECOM 2009)*, 2009, pp. 1–6.

[4] R. Maheshwari, J. Cao, A. Subramanian, F. Zarinni, and S. Das, "Adaptive channelization for high data rate wireless networks," Stony Brook University, Stony Brook, New York, Tech. Rep., 2009.

[5] "Ettus Research LLC," November 2010. [Online]. Available: http://www.ettus.com/

[6] "GNU Radio," November 2010. [Online]. Available: http://gnuradio.org/

[7] K. Liu, "Understanding the implementation of IEEE MAC 802.11 standard in NS-2," November 2010. [Online]. Available: www.cs.binghamton.edu/∼kliu/research/ns2code/note.pdf

[8] R. Calvo and J. Campo, "Adding multiple interface support in NS-2," January 2007. [Online]. Available: http://telecom.inescporto.pt/∼rcampos/ucMultiIfacesSupport.pdf

[9] G. Bianchi, "IEEE 802.11-saturation throughput analysis," *IEEE Communications Letters*, vol. 2, no. 12, pp. 318–320, December 1998.