Directional Wireless Mesh Network Design

James Martin

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-17
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-17.html

May 1, 2021

Directional Wireless Mesh Network Design

by

James Chillura Martin II

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor John Wawrzynek, Chair
Professor Robert Brodersen
Professor Paul Wright

Spring 2019

Directional Wireless Mesh Network Design

Abstract

Directional Wireless Mesh Network Design

by

James Chillura Martin II

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor John Wawrzynek, Chair

From a system point of view, this body of work examines the design and scaling of directional wireless mesh networks. Effort has been made to look at both the physical and medium access layers of these systems and what can be done to improve the capacity of these networks so that they may perform efficiently at very large scales. This work takes a forward looking view of these networks. While much of this work is based around millimeter wave radios (especially 60 GHz), it is not limited by it. Many of the ideas are independent of frequency and therefore should apply at both lower and higher carrier frequencies.

From the physical layer standpoint, I show that interference is a significant limitation to system capacity even for very large antenna arrays (1000s of elements), eliminating the idea of 'pencil beams' that do not cause or receive interference. Limiting and mitigating interference are investigated. To limit interference, I look at how the antenna array geometry may be altered and show that of the geometries I have examined, linear arrays perform best. To mitigate interference, I show that using receiver only adaptation and transmitter beam steering can be significantly better than using beam steering for both transmitter and receiver and helps to better approach the theoretical capacity limit.

At the medium access layer, I present a new protocol called listen-only scheduling. The principle idea behind it is that nodes in a mesh network each independently determine their listening schedules without transmission scheduling. The independent scheduling is enabled by the use of independent transmitters and receivers and could be used at any carrier frequency. It is particularly attractive for directional mesh networks because of deafness and exposed node issues. I show that theoretically it should be able to perform at least within 20–30% of the maximum theoretical throughput.

Lastly, future directions for this work are presented with some preliminary work. In particular, there are opportunities for cross-layer optimization between the physical, medium access, and network layer to avoid interference, increasing system capacity. One should be able to schedule in time using link management at the medium access layer and schedule in space and time by using routing at the network layer.

To Those Who Have Had a Positive Impact on my Life

Especially my mother.

Contents

List of Figures

List of Tables

Acknowledgments

There are numerous people who I would like to acknowledge who helped me to reach the point of finishing this dissertation.

First off and foremost, I must acknowledge my advisor, John Wawrzynek. Without John, I would have never started the PhD program at Berkeley. When I first met John, I was already working for him as a system administrator for RAMP, and he did not even know for a month. Over the years, I have worked with John as a system administrator, undergraduate researcher, systems engineer, and graduate researcher. However, it was only as a graduate researcher where our engagements became rather interesting. For the first few years of graduate school, my research followed up whatever was John's current interest. I worked on multi-core computer architecture for Chiplets; AVB and wireless latency for Terraswarm; free-viewpoint video for Unpad; and then finally settled on directional wireless mesh networks. This does not include the many side project discussions I have taken part in. The great thing about John was that his interests were varied from programming language to physiology to physics to culinary, and it was always easy to find a topic to discuss with him. Because of this, he also brought together an electic set of grad students that worked on often very different things. Many of my current interests were derived or inspired by these students including Greg Gibeling, Alex Krasnov, and Adam Megacz. Thank you John for being my advisor and bringing such a breadth of knowledge to my life.

Robert (Bob) Brodersen is really the whole reason why I ended up on this thesis topic to begin with. One day, during a group meeting with John, Bob is present and goes on about mesh networks, 60 GHz, and beam steering, discussing how he thinks this is the future. Even when I told him I knew almost nothing about antennas, radios, or really anything analog, he simply stated, "you can learn it." In the next week, he dropped off a few books for me to read. In the following years, Bob has proved to be a second advisor to me. I really appreciated his ability to argue and not give up until I finally presented enough evidence to convince him I was right and he might be wrong. Sometimes, we were both wrong. I learned a great deal through these discussions especially understanding what the fundamental problems or lack of understanding were on my or our part. Bob always treated me as an equal and did whatever he could to ensure that I was successful, volunteering whatever resources he had available to advance my research. He, in fact, was also the driving force that transitioned this work from a collection of papers, talks, code, and notes to a thesis. Thank you Bob for being a great advisor.

Adam Wolisz served as an advisor to me for anything that involved the medium access control or network layer. His experience in this area was immeasurable, and he seemed to always be up to date with what was going on in the wireless community. What was really great was how excited he would get about the numerous topics. I will never forget the many anecdotes that he gave to explain a decision that was made or to give an analogy to a phenomenon that was being observed. Adam also provided me a unique opportunity to work in Berlin over the summer, an opportunity that I count among my best experiences. At the many professional environments we attended, he would consistently make sure to connect me

with someone who might be working on something similar or sometimes just someone who was intellectually engaging. Adam's unique perspective compared to the people at Berkeley also gave me a larger perspective to research. Thank you Adam for all your assistance over the years, and I am sorry that I couldn't limit myself to three things about you.

To be honest, I am not sure how to start to thank Patrick (Shaobai) Li. I became friends with him almost instantly upon meeting. He was someone with whom I could discuss anything with. I was extremely fortunate to have met someone with an extremely large overlap of interests including, but certainly not limited to, programming languages, video games, movies, and philosophy. We regularly had coffee throughout graduate school and discussed whatever was on our minds. He would often throw a programming language question at me, and I would describe something about wireless that he did not know much about. However, even if he did not know at the time, he would take the time to actually understand any problem I threw at him, ask insightful questions, and offer useful suggestions. Nearly everything in this thesis, Patrick and I have discussed at some point. Thank you, Patrick, for everything. I hope we are still having coffee when we are old and gray.

Gregory Wright from Nokia Bell Labs was a consistent source of insight into just about any technical problem that I was researching. Given a problem, he would carefully consider the solution, sometimes coming back after a few hours having written in his notebook to help him understand the problem. Greg has read and offered comments on every paper I have submitted, often as a last minute request from me. On a personal level, I have enjoyed the many technical discussions we have had, not just about wireless but about Haskell, beer, and his many amusing stories. Thanks Greg for being both a friend and colleague.

Alexander Krasnov was the one person I knew that if I was having a difficult problem (no matter the topic!) he would know the answer or at least the right direction to go in. His first reply was almost always, "I don't know." However, upon further pressing, he would reveal the wealth of knowledge that has consistently surprised me. Alex also consistently taught me to go beyond my comfort zone including academically, socially, and physically. I have enjoyed the many discussions and road trips that we have had, and I hope that they continue.

I want to thank all of the administrative and engineering staff at the Berkeley Wireless Research Center that I have engaged with over the years. They have made some of the overhead of being a graduate student less trying.

Lastly, I would like to thank all of my fellow graduate students who have been a consistent source of inspiration and a general outlet for those times when you just need someone else to talk with.

Chapter 1

Introduction

Throughout the history of electric communication, there has been two dominant forms of communication, wired and wireless. Wired communication came first with the electric telegraph. However, wired communication is not capable of handling arbitrary mobility, and with the advent of the invention of the radio by Hertz and Marconi, wireless communication became possible to handle all forms of practical communication.

If one examines the evolution of wireless communication, one will notice that the carrier frequency and bandwidth has increased due to the developments of technologies such as vacuum tubes, transistors, and microprocessors. However, at each carrier frequency, there are presented new challenges. In the 30–300 GHz band or mmWave, antenna aperture sizes are on the order of millimeters. Because of small aperture size, various methods have been used to increase the amount of gain. However, they boil down to two methods: physical devices that focus a single antenna element increasing the gain (e.g. horn or parabolic antenna) and antenna arrays (multiple antenna elements).

Recently antenna arrays have become a viable option due to advances in technology. However, because there are many antenna elements, there is now a strong spatial component to communication, giving a new degree of freedom that is only recently being explored. This work starts to address these problems, specifically looking at the geometry of antenna arrays, optimal antenna array coefficients, and mesh topology medium access control. While this work focuses on millimeter wave, much of the analysis can be applied to directional networks in general.

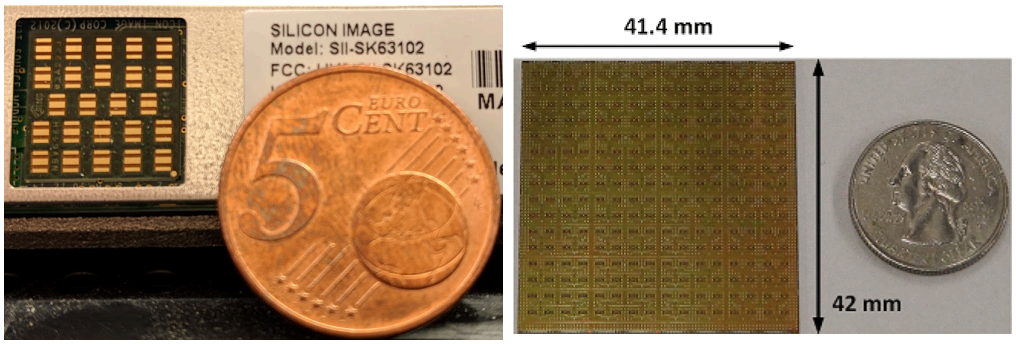
1.1 Motivation

The core motivation of this work is the ability to dynamically take advantage of spatial diversity. Millimeter wave bands enable large antenna arrays and therefore highly dynamic directionality. Mesh networks can be drastically improved by lowering the interference because of the ability to form nulls.

Millimeter wave

Millimeter wave bands offers a number of advantages over lower bands. Firstly, the Federal Communications Commission has made the 57–64 GHz spectrum available for unlicensed operation [16]. Similar spectrums are also available in other countries [10] [21] [32]. This is due in part because of high absorption in atmosphere in this band [20]. However, the high absorption offers an advantage of increasing spatial isolation. The fact that the band is unlicensed makes it commercially viable for many different applications. For example, there are many different protocol standards for the spectrum [30] [18] [6] [70]. 802.11ad is perhaps the most ubiquitous one. In addition, 7 GHz of bandwidth (slight variation of bandwidth depending on country) allows for very high throughput links because capacity is linearly dependent on the bandwidth.

Millimeter wave offers a significant advantage due to the relatively small size of an antenna element. An antenna element is typically proportional to the wavelength, λ. Sizes of $\lambda/2$ to λ are common. As the name implies, an antenna element in this band is on the order of millimeters. This offers an advantage in that it can be used with very small devices. However, a significant advantage can be had if the device in question is much larger such as cell phones or laptops. In this case, many antenna elements (10s-1000s) can be used as part of an array allowing for electronic steerability of a highly directional beam. Two examples are shown in Figure 1.1.



(a) 24-Element 60GHz Array (b) 16×16 60GHz Array [72]

Figure 1.1: mmWave Antenna Array Examples

High directivity offers a number of advantages. First off, as a transmission radiation pattern becomes more directional, the efficiency at the direction of interest increases because more energy is being transmitted in that direction and less in another direction. This efficiency is referred to as the transmission directional gain of an antenna. Having a directional receiver means that the receiver is subject to less other signals as the receiver's pattern becomes more directional in addition to also receiving higher gain. If one keeps the power fixed, then the gain of the antenna array increases linearly with the number of antenna elements. This is important as regulatory bodies (e.g. FCC) [16] often control the effective isotropic radiated power. However, there is no regulation on the receive gain. Therefore, if one scales

the power with the number of antenna elements, the overall improvement in receiver gain can be quadratic with the number of antenna elements. Therefore, if one has the same number of elements, n, for both the transmitter and receiver, the minimum improvement in received power is n^2. If one scales the power of the receiver as well, the overall improvement in received power is n^3.

In addition to the improvement in gain, because of directivity, less interference should be transmitted into the environment, and receivers should receive less interference. Less interference leads to overall improvement in data rates for all links. It also been shown in a study that, in the presence of human notion, narrow receive beams have a higher coherence time [57]. Lastly, with directivity, it should also be more difficult for other devices to eavesdrop on a particular link. The only tradeoff to all of these benefits is the loss of omnidirectional transmission, which is useful for when broadcast is desired or overhead needs to be minimized when directions of interest are not known a priori.

Mesh networks

A mesh network is a collection of nodes consisting of at least one transceiver each where the nodes may communicate with any of its neighbors. An example of a mesh is shown in Figure 1.2. In this figure, there are eleven nodes, and each of these nodes may communicate along any of these edges. Other topologies such as trees or rings are considered to be subsets of meshes.

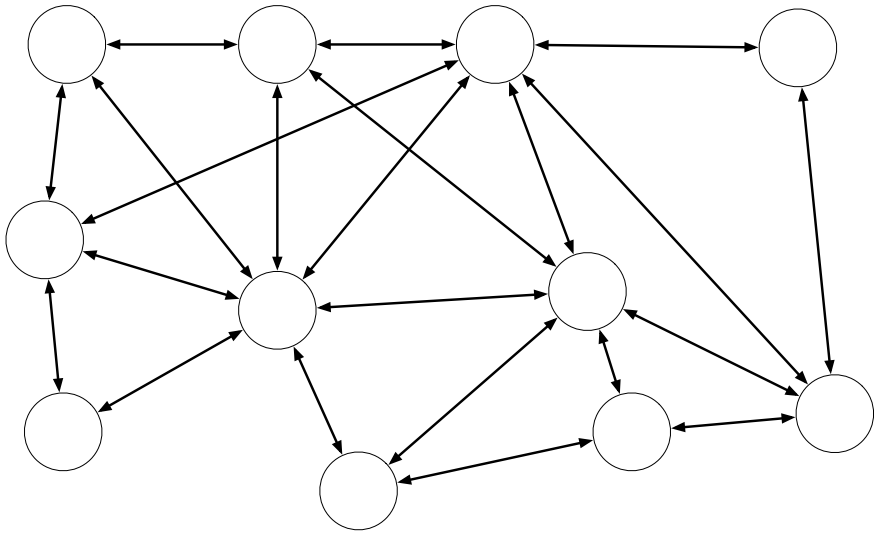


Figure 1.2: An example of a mesh network

There are many advantages offered by mesh networks. For sufficiently large networks, there is a great deal of redundancy. Should a link or node fail, the rest of the operational nodes may use alternative paths through the network due to the high number of links as well as nodes. Mesh networks also scale with the density of the network. Unlike other topologies,

when new nodes are added, the decision as to which nodes it may communicate with is not made a priori, but instead based on the connectivity available to it in its environment. This leads to a sharing of burden for traffic communication as opposed to a few select nodes handling most of the traffic and control.

However, traditionally, mesh networks have been shown to have challenges. The biggest problem comes from interference, which leads to use of shorter links, which leads to traffic congestion. A famous theoretical result showed that, in the limit, fully wireless networks' capacity scales with the square root of the number of nodes as opposed to linearly [26]. For this reason, mesh networks has been mainly viable for low data-rate applications such as smart meters [25] [63].

Millimeter wave mesh networks

Most wireless communication networks, at present, are a collection of spoke-and-hubs (i.e. one node services the rest of the nodes in the network). In recent years, it has become obvious that only a single wireless connection multiplexed among end devices and a base station is frequently insufficient, and multi-hop mesh configurations improve the coverage as well as reliability (such as Eero [19] and numerous followers). Multiple projects have demonstrated the efficiency of WLAN meshes for supporting internet access to urban and rural communities [5].

All classical mesh configurations suffer, however, from the inherent problem of omnidirectional wireless transmission—namely, the inter-link interference severely limiting the total system capacity [26]. The engineering community has therefore looked toward directional beams to mitigate interference. For example, Eero uses meshes in 5 GHz ISM band with 3–8 antennas, achieving beam widths of 30°–60°.

As mentioned earlier, what makes millimeter wave attractive is that many antenna elements can fit in a small physical space. Therefore, very narrow beam widths are possible. In addition, the number of nulls (or directions such that that gain is close to zero) is equal to $n - 1$, where n is the number of antennas. This means that as more antenna elements are added higher densities can be supported by nulling interference. Therefore, it should be possible to support much higher capacities than what was previously achieved with lower frequency mesh networks.

1.2 mmWave mesh networks

mmWave mesh networks consist of three abstraction layers: physical layer, medium access control layer, and network layer. The physical layer is concerned with optimally sending data for a single link. The medium access control layer pertains to the coordination of multiple potential communication links for a single node to multiple other adjacent nodes. Lastly, the network layer refers to the management of multiple nodes to form paths through the network. Metrics of mmWave mesh networks include:

- Latency: Time from a source to destination in a network.
- Throughput: Number of bits per second that the network sustain.
- Fairness: Division of resources to individual routes.
- Energy: Amount of energy required.

Physical Layer

The main issue for any link at the physical layer is how to maximize the capacity of a single link. Capacity, as defined by Shannon, is determined by the bandwidth, desired signal's power, the interference power, and the noise power [52]. The desired signal is the transmission signal of interest. Bandwidth is measured in Hertz and represents the range of frequencies that the signal occupies. Interference is other transmission signals that are not of interest. Noise is any other source of loss in the transmission of the signal that is not correlated with interference. The maximum capacity for a link will be such that the desired signal's power is maximized and the interference and noise are minimized.

Typical techniques at the physical for maximizing capacity at lower frequency attempt to maximize the signal and treat interference as if it were noise. In other words, techniques that are applicable for noise are also applicable for interference (e.g. forward error correction). Most techniques for maximizing capacity at all carrier frequencies attempt to maximize the signal to overcome typical sources of loss such as path loss (e.g. increasing antenna gain or transmission power) and multipath (e.g. channel estimation).

The difference at mmWave than at lower frequencies is that there are frequently a large number of antenna elements that can be individually controlled in amplitude and phase. There are two main reasons for this: the need for a large aperture and steerability.

The first, the need for a large aperture, is directly related to the fact that the antenna size is proportional to the carrier wavelength, λ. Typical antenna sizes are around $\lambda/2$. At mmWave, as the name suggests, the antenna sizes are on the order of a few millimeters. The direct consequence of this is that path loss can be significant due to the small aperture size. Therefore, increasing the aperture is highly desirable for links that need to work at high data rates for more than a few meters. In the past, horn and other directional antennas were used to take advantage of this [38] [29]. However, even though directional antennas were possible much earlier than large antenna arrays at mmWave, they had limited commercial applications because they were highly directional in a fixed direction and therefore were only able to be used in applications where it was unnecessary to steer the antenna such as in long distance backhaul [61]; it was acceptable to mechanically steer the antenna such as in radio astronomy [38]; or the size and cost of the directional antennas was okay for the application such as in radar [22].

With the advent of CMOS mmWave antenna arrays [2], it became cost effective to build many antenna elements with an overall small physical size. These antenna arrays enable communication links between a few meters to kilometers in a flexible manner due to the

electronic steerability via controlling each of the antenna elements in the array. Therefore, mmWave can be used for communication in applications such as cellular and mesh networks where the network topology at over a given period of time is not fixed.

There are many considerations for the physical layer when designing the CMOS including the antenna array, transceiver architecture, and packaging [17]. Commercial antenna arrays in mmWave for 802.11ad are typically planar on either silicon or a PCB and with a grounded backplane [23] [14] [69]. The number of antenna elements and the geometry, however, is not consistent. This body of work will start to address what an optimal geometry and number of elements may be.

Medium Access Control Layer

The medium access control (MAC) layer is responsible for flow control and multiplexing of the wireless medium. This includes the decision as to which link among the possible set of links is used as well as the discovery of new nodes in the network.

The major types of medium access control can broadly be separated into two categories: carrier sense and time scheduled. Carrier sense is perhaps the most popular control mechanism. Nearly every commercial WiFi device implements carrier sense. The main idea behind carrier sense is that a transceiver first listens to the medium to determine if another transceiver is currently transmitting. If it hears nothing, then it proceeds to transmit. If it does here another signal, then the transceiver waits a period of time, often determined by random exponential backoff, before attempting to listen and then transmit [39, 15]. Due to the possibility of multiple transceivers attempting to transmit simultaneously, the mechanism requires a method to know if retransmission is necessary, often implemented using an acknowledgement. The main advantage of carrier sense is that if the medium is not being used heavily then a transceiver is able to start transmission almost immediately most of the time and is unlikely to be interrupted. If the wireless medium is being used heavily, then there enters an issue where nodes must wait longer and longer in order to decrease the probability of a collision, leaving many time gaps without transmission reducing the effective utilization of the medium. The other disadvantage in the technique described thus far is that there is an implicit assumption that every node is able to listen for any other node transmission that may cause a collision. Unfortunately, there are many scenarios where one is unable to determine if others are transmitting leading to the classic hidden node problem [62, 24]. An example of the hidden node problem can be seen in Figure 1.3. In this example, Node A and C would like to communicate to Node B, but due to fading, are unable to determine if the other node is already communicating. A popular workaround for the hidden node problem is to use request to send/clear to send mechanism (RTS/CTS) [36]. A node initiates a communication by using a small packet indicating how long it requests to transmit. The recipient of the communication sends a clear to send protocol with the same duration if the node is not busy. In this way, if every node has the same range, then all nodes that can cause collision will hear at least one of the two transmissions.

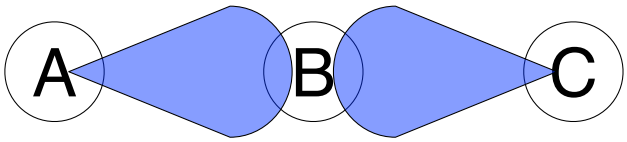


Figure 1.3: Hidden Node Example: Node A and C are attempting to communicate to Node B and cannot detect each other due to fading.

At mmWave frequencies, there are new difficulties that make carrier-sense more difficult. The difficulties arise from the fact that in mmWave it is very rare that one would choose to use an omnidirectional signal due to the issues mentioned earlier. This implies that the emission patterns are most commonly directional. Therefore, there are two new challenges that carrier-sense needs to deal with: deafness and exposed node.

Deafness occurs when a node cannot detect another node transmitting because it is currently listening in a different direction. An example is illustrated in Figure 1.4. In this example, imagine that Node A is transmitting to Node C. Node B is listening in the direction toward Node C and therefore cannot tell that Node A is transmitting. Node B is deaf to Node A. Note that in this scenario an easy fix would be to listen in the opposite direction of Node C, toward Node A. However, in general, one must listen in all neighbor directions to determine if it is okay to transmit to Node C.

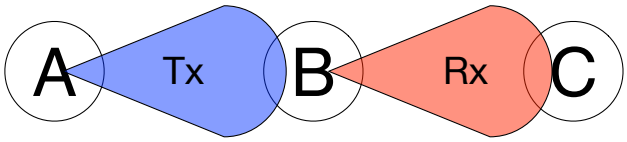


Figure 1.4: Deafness Example: Node B cannot detect that Node A is transmitting.

The exposed node problem is also a challenge unique to directional transmissions. It is caused by detecting a node transmitting and choosing to not transmit even though the ongoing and new transmission can coexist without interference. An example of the exposed node problem is shown in Figure 1.5. In this example, Node C is communicating to Node B. Node A would like to communicate to D. However, Node A detects Node C's transmission and chooses not to transmit because it is believed the medium is being used. However, due to directionality, these two transmissions can coexist without interference.

An alternative to carrier sense is time scheduled, often called time division [40]. One way to think of carrier sense is that the schedule of transmissions is created implicitly. Time scheduled makes the schedule of transmissions explicit rather than implicit. Abstractly, each node has a schedule that determines when it may use the transmission medium. The schedule may either be static or dynamic. Time scheduling, generally, leads to efficient use of the wireless medium when many nodes wish to transmit. Unfortunately, it is usually inefficient at low utilization due to a node needing to wait until an opportunity to transmit appears even though the wireless medium may be free. Dynamic scheduling can mitigate

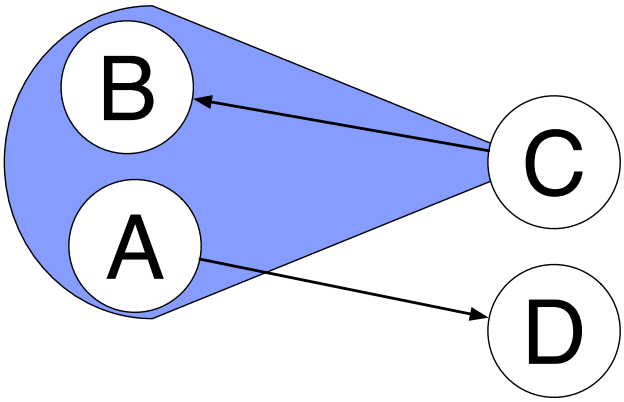


Figure 1.5: Exposed Node Example: Node C is transmitting to Node B. Node A would like to transmit to Node D but cannot due to detection of Node C's transmission. Both transmissions, though, can coexist.

this efficiency loss to some effect. However, it is often quite difficult to determine an optimal schedule due to the lack of knowledge of when nodes will decide to transmit. If one had perfect knowledge of when nodes will transmit, one could determine an optimal schedule, albeit with a non-trivial amount of computation, leading to optimal efficiency in terms of utilization of the network as well as minimizing delays through the network. Because of these inefficiencies, hybrids of both carrier sense and time division have been proposed [53].

Time scheduled offers a few advantages compared to carrier-sense, especially for mmWave. Any challenge related to sensing is completely avoided, specifically the hidden node, deafness, and exposed node problems. This makes it attractive for mmWave. In addition, because everything is scheduled power-saving is easily implemented as adjacent nodes will not attempt to transmit when a node is powered down. However, there is another challenge, in addition to the inefficiencies mentioned at low utilization, that carrier-sense does not have but time scheduled does. Nodes must coordinate with each other to determine schedules leading to additional communication and computation overhead. In addition, nodes must also be time synchronized in order to be able to schedule.

There exists two other popular medium access control schemes: frequency-division and code-division. Frequency-division, as the name implies, separates transmissions into different frequency bands [13]. This schema is effective if one has enough bandwidth to support each node independently. If not, another protocol such as carrier sense must also be employed. Code-division relies on each node using codes to make each node's transmission orthogonal (or statistically orthogonal) [64]. The general technique modulates a given signal to a much larger bandwidth (hence, why it is called a spread-spectrum technique). Code-division makes the tradeoff of significantly using more bandwidth than needed by a given signal leading to inefficiency in spectrum to gain orthogonality. Both of these schemes, however, do not appear to present new challenges at mmWave.

In addition to control of the physical transmission medium, the MAC layer is often

the layer where the transmission specifications are given. If one examines what the IEEE 802–2001 standard [31] encompasses in addition to what has already been discussed, they include:

- Frame delimiting and recognition
- Addressing of destination stations
- Conveyance of source-station addressing information
- Transparent data transfer of LLC PDUs (or equivalent)
- Protection against errors

Generally speaking, most of these specifications relate to the packet format and the semantics of the control information of the packet. Control information includes things such as modulation (e.g. BPSK or 16QAM), coding scheme (e.g. forward error correction scheme and code rate), and the packet type (e.g. control or data packet). The details of these schemes are relatively unimportant as they can vary quite significantly from protocol to protocol and are relatively orthogonal to mmWave and directionality.

Network Layer

The network layer is responsible for routing, packet forwarding, and addressing. Routing is the process of determining a path through two or nodes in the network.

Unlike degenerate meshes such as trees, a general mesh topology does not necessarily have predetermined routing before the network is created. This may be due to the network being ad hoc, the topology changes dynamically with time and therefore is unknowable indefinitely. It may also be due to the quality of links in the network being a function of time due to the environment (such as moving people). Therefore, in general, the current state of the topology must be discovered in order to determine the routes.

The simplest strategy for delivery of packets is actually not to do any routing using a scheme called flooding. A source sends its packet to all its neighbors. Those neighbors then send the packet received. This is done recursively until all nodes in the network have received the packet. The advantage of this strategy is that it is very simple and possibly has very high reliability as every edge and every node in the topology is used. Therefore, with a high probability the intended recipient will receive the message. However, the overhead is significant, proportional to the total number of nodes and edges in the entire network. Because of its very high overhead but potential for high reliability, flooding is only seen when the environment and topology is highly dynamic and unpredictable.

If one can assume that there is at least some stability to a given source and destination node in a network, routing may be employed. In this case, when the topology has a high variance in time (but not too high!), source-initiated routing (sometimes called reactive or on-demand) may be employed. In source-initiated routing, the source node floods the network

with route requests. Upon the route request reaching the destination, the destination node uses the discovered route to issue a route reply back to the source. Examples of source-initiated routing include DSR [35] and AODV [48]. Figure 1.6 illustrates an example of Dynamic Source Routing (DSR). At each hop of the route request, the previous hops are kept as part of the route request. If a node is visited twice, the route request is not forwarded.

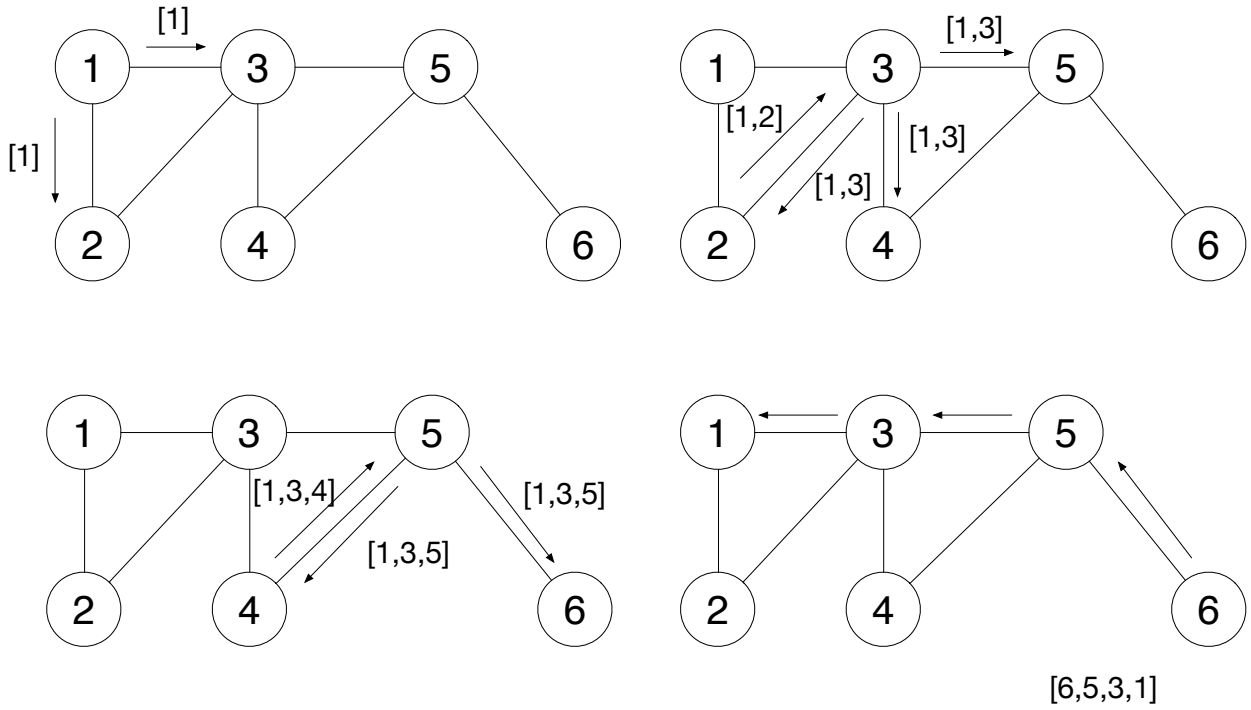


Figure 1.6: Example of Dynamic Source Routing: Route from Node 1 to 6.

The other popular type of routing protocols are referred to as table-driven (or proactive). In this scenario, routing tables are maintained with up-to-date information of routes. Changes in the network cause route updates, which are broadcast to remote nodes. Table-driven routing is well suited for networks that are static for long periods of times. Examples of table-driving routing include OLSR [33] and DSDV [49]. As an illustration, in Optimized Link State Routing (OLSR), each node periodically broadcasts the status of its links, Figure 1.7. A broadcast from a given node, node S in the figure, is only forwarded by its multipoint relays, nodes C, D, and E. Multipoint relays for a given node are its neighbors such that it has complete coverage of all of its two-hop neighbors.

In addition to source-initiated and table-driven, there are other routing algorithms. Hybrids of source-initiated and table-driven exist to handle the highly dynamic and relatively static parts of a network, respectively, e.g. [27]. In location-aware routing, nodes are aware of their location in the network, and the destination is a coordinate. Routing is done by computing the vector through the next node such that the packet makes forward progress, e.g. [41, 37]. Multipath routing is a routing algorithm that uses multiple paths from source

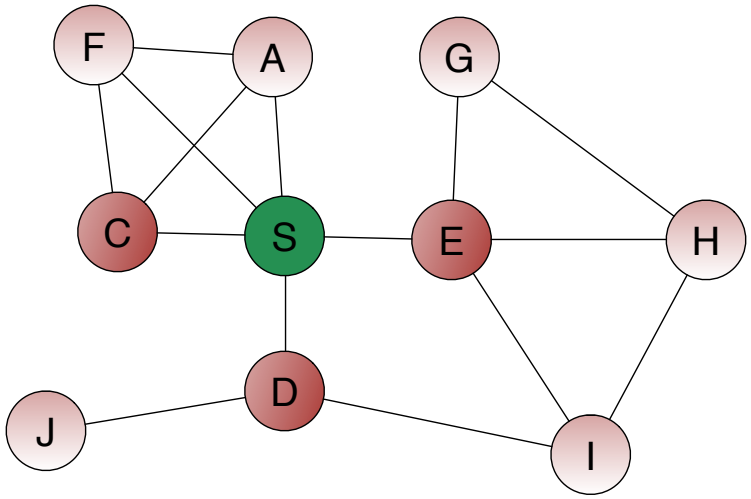


Figure 1.7: Example of Optimized Link State Routing. Nodes C, D, and E are the multipoint relays of Node S.

to destination to provide higher reliability, e.g.[]champ,smr. Hierarchical routing can be used similar to what appears in the internet today, where a tree is imposed inside of the mesh topology. These and many other routing algorithms may be found in various surveys [9, 54].

Routing in directional mesh networks proves to be more difficult. One of the main sources of difficulties is that, due to the lack of omnidirectionality, there is no $O(1)$ broadcast mechanism. As described earlier, many of the techniques described earlier relied on broadcast (or flooding) in order to discover paths through the network. In the presence of directionality, it is possible that these broadcasts will not successfully transmit to all nodes it could if an omndirectional broadcast and receive had been used.

To overcome this inability to broadcast, it has been proposed to divide up an omnidirectional broadcast into directional sectors and sweep [12]. This scheme is employed in 802.11ad. However, this approach is suboptimal if the receivers are also directional as it is possible for a route request to be missed due to the receiver's directional sector not intersecting with the transmitter's directional sector. Therefore, unless one is willing to provision time to guarantee reception, the alternative is to either use out of band communication such as 802.11ac or to use omnidirectional receiving. In Section 5.2, an alternative is presented that leverages a combination of scheduling and sectoring to simulate omnidirectionality.

1.3 Antenna Arrays

An antenna array consists of multiple antenna elements that work together to transmit or receive.

In mmWave, as mentioned earlier, the most common configuration is of planar antenna arrays with homogeneous antenna elements (i.e. the same antenna element response for each

element). Each individual antenna response is less than 180° in both azimuth and elevation. The limits of the individual antenna element response dictate the limits of the antenna array response as well. This means that, for example, if your antenna elements have no response beyond n degrees, it will also be true that your antenna array will have no response beyond n degrees.

The key attribute that antenna arrays take advantage of is that when two or more identical waveforms (except possibly a change in phase) are transmitted from different physical locations the waves interact with each other such that, depending on the physical location, they add constructively (increasing the amount of power) or destructively (removing power). While there are multiple different antenna array architectures and geometries, all of them consist of a set of antenna arrays with the same waveform given to some subset of antenna elements. Each antenna element most often also has a complex weight associated with it. The phase of the weight is used to steer the beam or adjust the overall pattern where the waveforms constructively and destructively add. The amplitude of the weight determines the overall amplitudes of the radiation pattern. If this complex weight can be controlled electronically, this leads to the ability to adjust the radiation pattern dynamically. For more on this, see the Interference Mitigation (Chapter 3). The specific algorithms can be found in Section 3.3.

Antenna Array Gain

Formally, if there are N antennas and assuming three-dimensional space, given a particular direction $\vec{v} \in \mathbb{R}^3$, the gain of an antenna array is a function of the position, $\vec{a_i} \in \mathbb{R}^3$, and weight, $w_i \in \mathbb{C}$, of each antenna element $i \in [1, N]$.

$$arrayGain(\vec{v}) = \left| \sum_{i=1}^{N} w_i e^{j\vec{a}_i \cdot \vec{v}} \right|^2 \tag{1.1}$$

From the equation above, it should be obvious that if the power scales with the number of antennas, (i.e. $\forall i, |w_i| = 1$), the array gain scales quadratically. As mentioned earlier, regulatory bodies often limit the effective isotropic radiated power of the transmitter, and therefore the amplitude of each weight will scale inversely with the number of antenna elements, (i.e. $\forall i, |w_i| = 1/N$), leading to linear scaling of gain with the number of antenna elements.

Radiation Pattern Characteristics of Uniform Arrays

Because many mmWave antenna arrays are uniform in spacing, it is good to examine some of the characteristics of uniform antenna arrays. The antenna array pattern consists of a mainlobe, two or more sidelobes (depending on the number antenna elements), and nulls between the lobes. Figure 1.8 illustrates a linear array radiation pattern with sixteen elements and $\lambda/2$ spacing and serves as a reference for terminology. Typically, the mainlobe

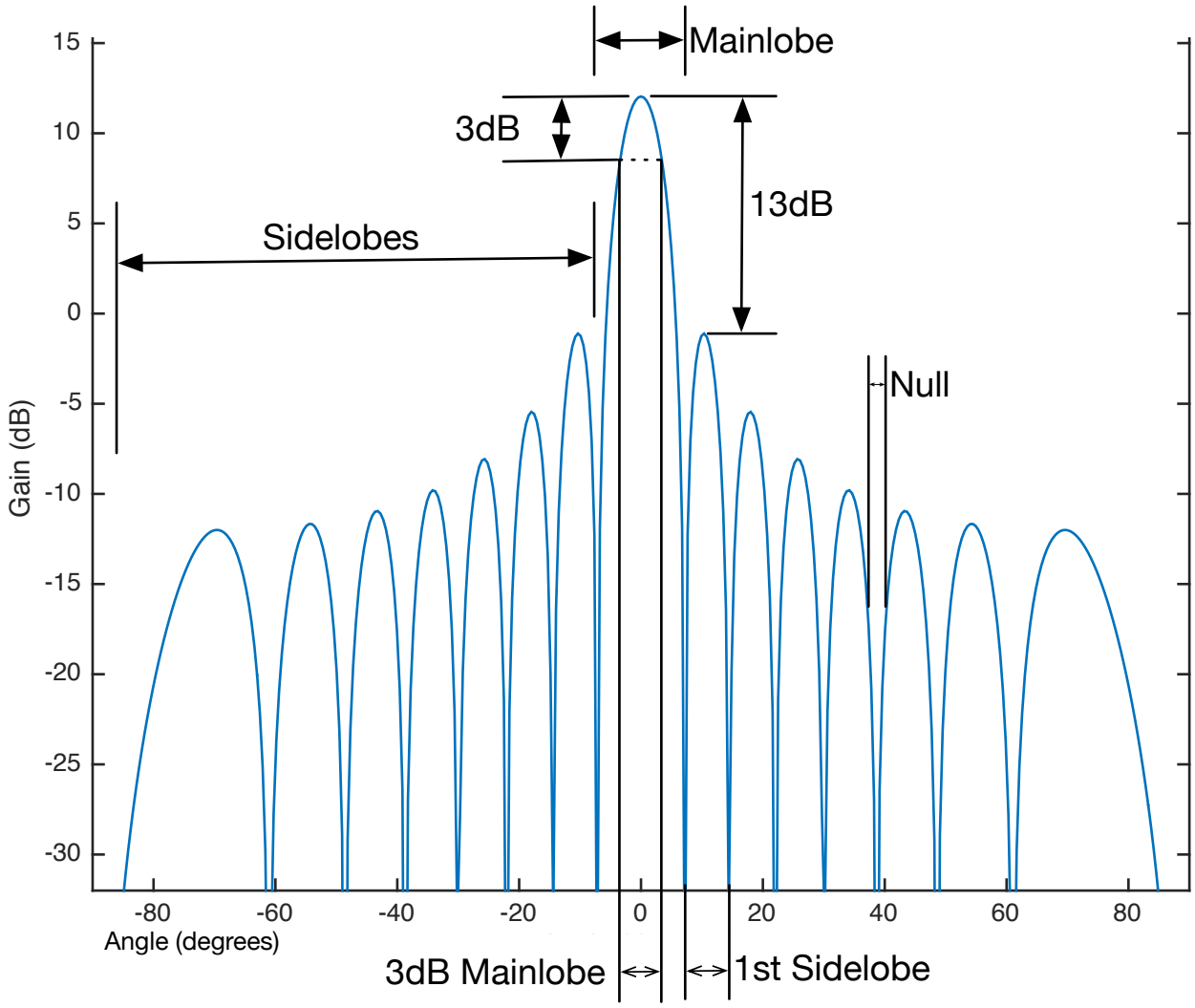


Figure 1.8: Uniform Antenna Array Radiation Diagram

is defined either to be the highest gain point plus everything between the first two nulls or what is within 3 dB of the highest gain point, typically called the 3 dB beam width. Given n antenna elements with $s\lambda$ spacing, the 3 dB width of a uniform antenna array is:

$$2\left|\arccos\frac{1}{s\cdot n}-\frac{\pi}{2}\right| \tag{1.2}$$

Sidelobes are every lobe that is not the mainlobe. For uniform arrays with equal amplitude consisting of at least five antenna elements, the first sidelobe's gain is 13 dB less than the mainlobe's gain. The gain of the additional sidelobes are a function of the antenna spacing (see next section). Nulls are defined to be the points where the gain is equal to zero (i.e. $-\infty\ dB$). Additionally, the number of independent nulls that may be manipulated is

equal to the number of antenna elements minus one. This is relevant for the purposes of adaptation (Chapter 3).

Figure 1.8 illustrates a linear array radiation pattern. However, because mmWave antennas are typically built on a two-dimensional planar surface, two-dimensional arrays are possible and common. Two-dimensional arrays create a structure in three-dimensional space. Figure 1.9a shows the same 16 element linear array visually in three dimensions. Rectangular arrays look identical to linear arrays in azimuth and elevation. However, in three dimensions, the fan structure is gone, and instead the classic three-dimensional beam pattern is seen. Figure 1.9b shows a 4×4 element array with $\lambda/2$ spacing.

For non-rectangular two-dimensional arrays, the radiation pattern will vary. A uniform hexagonal array operates similarly to rectangular arrays in terms sidelobe levels. Figure 1.9c shows a 19 element hexagonal array with $\lambda/2$ spacing (19 in order to form a hex grid). Uniform circular arrays are similar to rectangular arrays. They are essentially rectangular arrays with depopulated antenna elements. However, uniform circular arrays have higher sidelobes to mainlobes, 8 dB. Figure 1.9d shows a 16 element circular array with a radius of $\frac{\lambda}{4\pi}$ giving a $\lambda/2$ spacing. Note that in this figure, the other half of the response (i.e. the elevation below zero) is shown even though it should be zero. This body of work will focus on mainly rectangular arrays of $\lambda/2$ spacing.

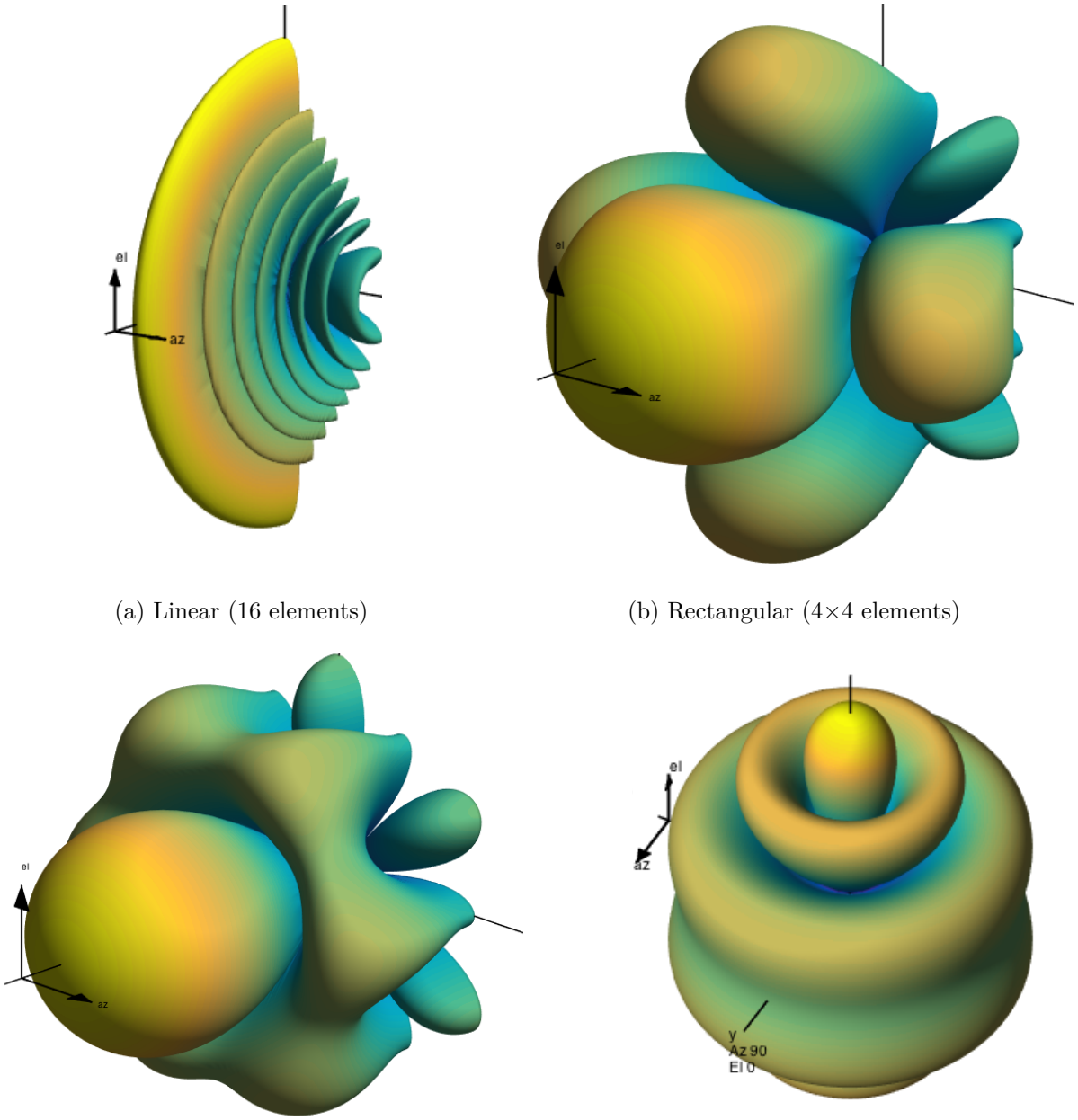
Antenna Spacing

Typically, antenna elements are spaced, s, between half a wavelength and one wavelength apart, precisely $\lambda/2 \leq s < \lambda$. Spacings below half a wavelength give the same amount of gain but with a wider mainlobe. Therefore, it is always suboptimal to use an antenna spacing less than half a wavelength. Spacings that are equal to or greater than λ have what are called grating lobes, sidelobes of equal strength to the mainlobe. This is often undesirable as there are essentially multiple mainlobes even though the primary purpose of the array is to establish a single link. Examples of different spacings for a uniform linear antenna array of sixteen elements is shown in Figure 1.10.

Using half wavelength spacing, one can show that every sidelobe starting from the mainlobe decreases in gain. The tradeoff being made by increasing the spacing beyond half of a wavelength is that the mainlobe becomes narrower and the sidelobes become higher. As an example, if one examines Figures 1.10c and 1.10b, one will note that the sidelobes on the extreme ends are approximately 3 dB higher for the 0.75λ case vs the 0.5λ case. As the spacing increases toward λ, the extreme sidelobes will increase toward the same gain as the mainlobe, Figure 1.10d.

Recovering Individual Antenna Element Response

Transceiver architectures can vary greatly. However, very typically, each antenna element will have phase control if not amplitude control. Access to the individual antenna element's outputs may be impossible as the summation may be done in analog. The individual antenna



(a) Linear (16 elements)

(b) Rectangular (4×4 elements)

(c) Hexagonal (19 elements)

(d) Circular (16 elements)

Figure 1.9: 3D Radiation Patterns (Logarithmic)

element's outputs are of interest for techniques that attempt to mitigate interference and maximize SINR by controlling the phase and amplitude of each element. However, there are a number of ways to recover the individual antenna element responses. The most obvious way is, if you have amplitude control, to individually turn on each element. While this may work, the gain of a single element is very small and therefore subject to more noise. Alternatively, one may apply a series of coefficients from a non-singular matrix such that the entire aperture of the antenna array is used. One known matrix is the Hadamard matrix which relies on having available only one bit of phase, 0 and π, for antenna arrays that have number of elements equal to 2 or a multiple of 4 [58] [28]. If the number of antenna elements is not equal to 2 or a multiple of 4, then a complex Hadamard matrix may be applied, however, more bits of phase is necessary [59].

To recover the response for each element, $\vec{x}$, take each row of the non-singular square matrix, $\mathbf{M}$, of size $n \times n$ that was chosen, apply the coefficients from that row to the antenna array, and measure the total complex response of the array where n is the number of antenna elements. From this, you will end up with a vector $\vec{g}$ of length n.

$$\vec{g} = \mathbf{M}\vec{x} \tag{1.3}$$

Because the matrix chosen is non-singular, an inverse exists. Therefore, to recover the response, $\vec{x}$, one simply needs to multiply $\mathbf{M}^{-1}$ to $\vec{g}$.

$$\mathbf{M}^{-1}\vec{g} = \mathbf{M}^{-1}\mathbf{M}\vec{x} = \vec{x} \tag{1.4}$$

1.4 Notes on Syntax and Pseudocode

All algorithms in this dissertation are assumed to follow lazy semantics (though not necessarily lazy evaluation). For most of this dissertation, algorithms will be described in mathematical notation to keep it as broad audience as possible. However, pseudocode is used when the algorithm is iterative or recursive in nature.

Every algorithm described using pseudocode has a type signature. For example, a function, f, that takes as input a real number and an integer and outputs a boolean would be written as

$$f :: \mathbb{R} \to \mathbb{Z} \to \mathbb{B} \tag{1.5}$$

Functions may also be higher-order and have parametric types. Syntactically, parametric types are described using traditional type lambda calculus syntax and the parameters themselves are always lower case. The classic *map* function's type signature is given below.

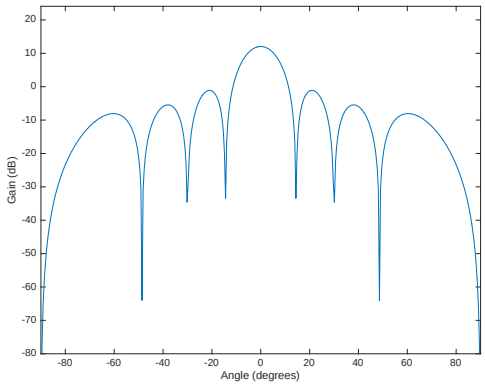
$$map :: \forall a.(a \to b) \to [a] \to [b] \tag{1.6}$$

The following types and their syntax are used in the paper. Readers familiar with Haskell or other functional programming languages should have no issues. Lists are a parametric type and are either the empty list or an element of the parametric type concatenated with a list.

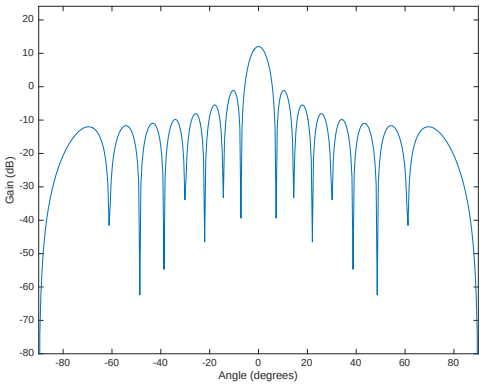
```
type [a] = [] | a : [a]
```

The `Maybe` type is a parametric type. It is either `Nothing` or `Just` an element of the parametric type.

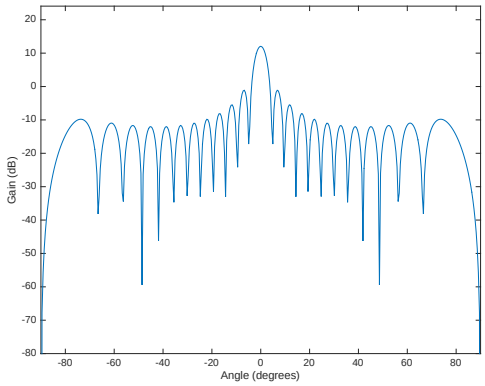
```
type Maybe a = Nothing | Just a
```



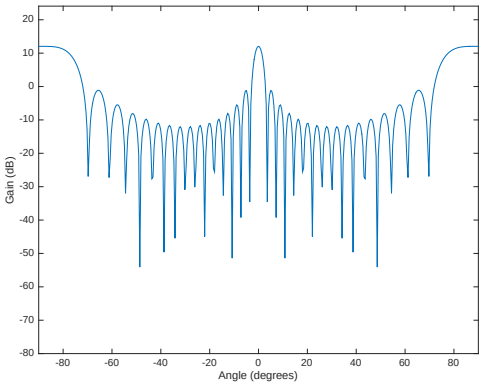
(a) 0.25λ



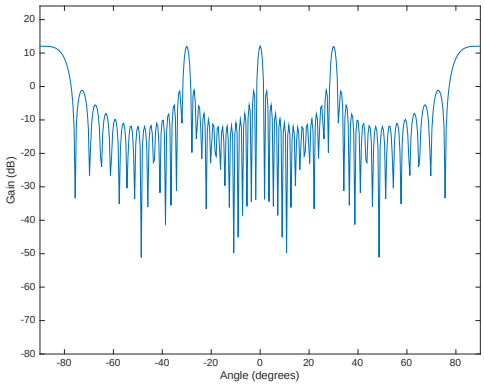
(c) 0.5λ



(b) 0.75λ



(d) λ



(e) 2λ

Figure 1.10: 16 Element Uniform Linear Antenna Array — Various Spacings

Chapter 2

System Model

For the purposes of simulation and analysis, it is nice to have a consistent model. The system is comprised of a network consisting of nodes with one or more antenna arrays where each antenna array is configured to be either a transmitter or receiver. The antenna array model is based on a standard electromagnetic wave model that assumes the incoming wave is planar, and there is no antenna coupling [47].

The rest of this body of work will refer to sections of this model. The Interference Mitigation chapter uses everything except there is no packet forwarding, so queues are not relevant. The Antenna Array Geometries chapter is based on this model but makes simplifying assumptions to speed up the computation. Listen-only scheduling only employs the node and traffic model.

2.1 Antenna Array

An antenna array consists of a set of antenna elements, $\mathbf{E}$, in a two dimensional plane that can form a three dimensional beam. For each antenna element $i \in \mathbf{E}$, the antenna element has a position, $\vec{a}_i$, and a coefficient, w_i. The position of the element is a three dimensional vector, $\vec{a}_i \in \mathbb{R}^3$. The coefficient is represented as a complex number for amplitude and phase of the antenna element, $w_i \in \mathbb{C}$. The coefficients are constrained to have a maximum amplitude of one, $\forall i \in \mathbf{E}, 0 \leq |w_i| \leq 1$. The beam coefficients for an antenna array is a $1 \times |\mathbf{E}|$ matrix given as $\mathbf{W} = \{w_1; w_2; \ldots; w_{|\mathbf{E}|}\}$. The gain of an antenna element is given as a cosine element:

$$antGain(\theta, \phi) = \begin{cases} \cos^2(\theta - \pi/2)\cos^2(\phi) & 0 \leq \phi < \pi \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where θ and ϕ are spherical coordinates by ISO convention. Because many mmWave antenna arrays have a grounded backplane, the response 'behind' the antenna is zero. The gain of

the antenna array at direction, $\vec{v} \in \mathbb{R}^3$, is given as:

$$gain(\vec{v}, \mathbf{W}) = antGain(toS(\vec{v})) \left| \sum_{i=1}^{|\mathbf{E}|} w_i e^{j\vec{a}_i \cdot \vec{v}} \right|^2 \tag{2.2}$$

where toS is a function from a vector to spherical coordinates. Antenna coupling and polarity are not used in calculating the gain.

2.2 Node

A node consists of at least one transmitter and one receiver. Each transmitter and receiver is able to operate simultaneously and independently. A node has no volume and therefore does not cause fading or reflections, but it does have a fixed position in three dimensional space. Nodes do not have any mobility. An antenna array is assigned to each transmitter and to each receiver (e.g. there are two antenna arrays if there is one transmitter and one receiver).

Each antenna array has an orientation of $\vec{v_o} \in \mathbb{R}^3$. For simplicity, the antenna element positions are considered independently of the orientation. Antenna elements exist in the YZ-plane with the normal of array being defined as $\{1, 0, 0\}$, corresponding to the spherical coordinates ($\theta = \pi/2, \phi = 0$). To calculate the gain for this antenna array, the incident ray, $\vec{v_i}$, must be normalized to be able to use Eq. 2.2. Assuming $\vec{v_i}$ and $\vec{v_o}$ are unit vectors, this normalization is defined as:

$$normalize(\vec{v_i}, \vec{v_o}) = rotate(\arccos(\vec{v_o} \cdot \vec{v_i}), \vec{v_o} \times \{1, 0, 0\}, \vec{v_i}) \tag{2.3}$$

$rotate(\theta, \vec{k}, \vec{v})$ is the Rodrigues' rotation formula [51].

Each node has its own clock allowing for time synchronization. In addition, there exists an output queue for every neighbor of a node 2.1. This requirement prevents the possibility of head of line blocking. When there is a possibility for multiple transmissions to different neighbors, round robin arbitration is used.

2.3 Channel

The channel between a transmitter and receiver is a multipath channel in a free space medium. The only modeled fading is free space path loss.

The power transmitted between a transmitter and receiver is given as a modified version of Friis' Formula to account for antenna array gain and additional sources of loss. For a given transmit direction, $\vec{v}_t$, and receive direction, $\vec{v}_r$, the total gain between the transmitter and receiver is:

$$totGain(\vec{v}_t, \mathbf{W_t}, \vec{v}_r, \mathbf{W_r}) = gain(\vec{v}_t, \mathbf{W_t}) gain(\vec{v}_r, \mathbf{W_r}) \tag{2.4}$$

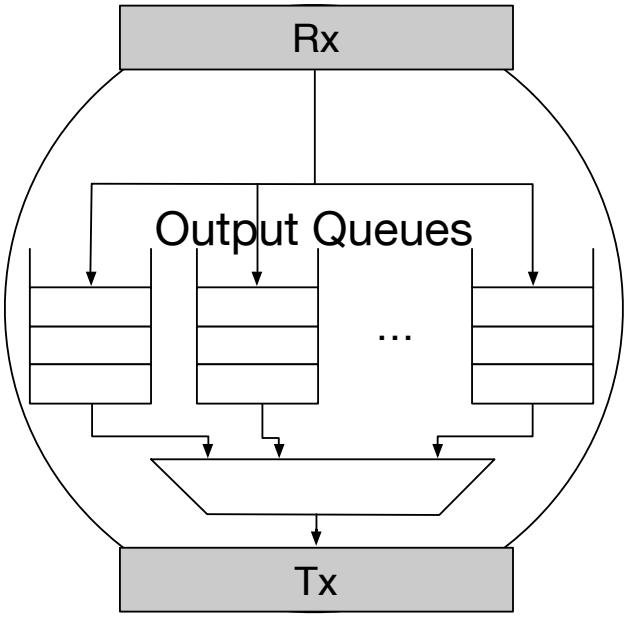


Figure 2.1: Node Model. An output queue for every neighboring receiver.

Reflections are accounted for using a constant multiplicative loss, $reflAtten$, representing a statistical average loss when reflecting from a surface. Unlike traditional reflection equations, the reflective loss is not a function of either material, angle, polarity, or roughness. Only specular reflection is taken into account. Diffraction is not modeled. A reflective loss is a function of the number of bounces, b, that occurred before reaching the receiver assuming that all surfaces are of the same material.

$$refLoss(b) = reflAtten^b \tag{2.5}$$

The total loss for a given path between a transmitter and receiver is a function of the distance, $dist$ in meters, of the path (path loss) and the number of bounces, b, the path takes:

$$totLoss(dist, b) = implLoss \cdot refLoss(b)\left(\frac{\lambda}{4\pi dist}\right)^2 \tag{2.6}$$

where λ is the wavelength in meters and $implLoss$ is a constant that represents the loss due to hardware.

For convenience, a path i between a transmitter and receiver, $p_i^{t,r}$, is defined to be a tuple:

$$p_i^{t,r} = (\vec{v}_t, \vec{v}_r, dist, b) \tag{2.7}$$

The total receive power for a given path, p, for a given constant transmit power, txP, is:

$$RxPower(p_i^{t,r}, \mathbf{W_t}, \mathbf{W_r}) = txP \cdot totLoss(dist, b) \cdot totGain(\vec{v}_t, \mathbf{W_t}, \vec{v}_r, \mathbf{W_r}) \tag{2.8}$$

The signal power between a transmitter, t, and receiver, r, is considered to be the maximum receive power of all the given paths between the two. This is the best signal power that can be achieved for single carrier, single user.

$$SignalPower(t, r, \mathbf{W_t}, \mathbf{W_r}) = \max_i RxPower(p_i^{t,r}, \mathbf{W_t}, \mathbf{W_r}) \tag{2.9}$$

The interference power between a transmitter and receiver is the sum of all paths between the transmitter and receiver:

$$InterfPower(t, r, \mathbf{W_t}, \mathbf{W_r}) = \sum_{i=1}^{\infty} RxPower(p_i^{t,r}, \mathbf{W_t}, \mathbf{W_r}) \tag{2.10}$$

The noise power, σ^2_{noise}, is the product of four constants: noise figure, $noiseFig$; room temperature, $temp$; bandwidth, bw; and Boltzmann's constant, k:

$$\sigma^2_{noise} = noiseFig \cdot bw \cdot temp \cdot k \tag{2.11}$$

The SINR for a receiver, r, with an intended transmitter, t', is then the signal power divided by the noise power and all interference power.

$$\text{SINR}(t', r, \mathbf{W_T}, \mathbf{W_r}) = \frac{SignalPower(t', r, \mathbf{W_{t'}}, \mathbf{W_r})}{\sigma^2_{noise} + \sum_{t \in T, t \neq t'} InterfPower(t, r, \mathbf{W_t}, \mathbf{W_r})} \tag{2.12}$$

Capacity for a link is calculated using Shannon's capacity formula:

$$Capacity(t', r, \mathbf{W_T}, \mathbf{W_r}) = bw \log_2 (1 + \text{SINR}(t', r, \mathbf{W_T}, \mathbf{W_r})) \tag{2.13}$$

2.4 Network

A network consists of multiple nodes in a three dimensional space. All nodes in the network are homogeneous. They have the same number of antenna arrays, all with the same geometry. The topology, what nodes can communicate to other nodes, and the geometry, the physical positions of the nodes, of the network is fixed for a given experiment.

Traffic from sources in the network is generated on a slot level, corresponding to fixed units of time, using a Pareto ON/OFF distribution.

$$\text{PDF} = \frac{\phi x_m^{\phi}}{x^{\phi+1}} \text{ for } x \geq x_m \tag{2.14}$$

ϕ is chosen to be 1.5 because of its common usage in the traffic simulation community. Each source in the network has two x_m, an OFF and an ON. The ON is the number of consecutive slots that the source is active. The OFF is the number of consecutive slots that the source is inactive before becoming active again. Each node is seeded with a different random seed. However, both ON and OFF are the same for all source nodes. The type of source (e.g. packets or active power) is dependent on the study.

Chapter 3

Interference Mitigation

This chapter examines the impact of interference on capacity for antenna arrays under an idealized scenario of networks in a rectangular cuboid where reflections are only from the walls of the cuboid. Ray tracing is used to model reflections. Different networks are considered with nodes consisting of large numbers of antenna elements. A distributed adaptive receiver algorithm is shown to improve capacity up to 60% compared to no adaptation at all.

3.1 Introduction

In the scientific community, there has been an intuition that has been propagated stating that beamforming with very large antenna count arrays can be treated as pseudo-wires [46]. Pseudo-wires would enable infinite scaling because wireless links would be essentially interference free enabling multiple simultaneous links.

This chapter challenges the pseudo-wire assumption by examining if realistic antenna arrays with sidelobes can support this assumption. In contrast to previous work, the full antenna array response is accurately model for large network deployments. Model-based simulation demonstrates that unfortunately this is not the case and networks with large arrays (100s of antenna elements) still suffer from significant interlink interference, reducing the capacity up to 50%. This interference results from the existence of sidelobes in the spatial patterns of the antenna array.

One of the major contributions of this work includes showing how capacity scales with the number of nodes as well as the number of antenna elements. In addition, given the discovery of the huge impact of interference on capacity, a fully distributive technique is presented and contrasted with doing no adaptation. The technique consists of using beamsteering for transmitters and receivers performing adaptation using minimum mean squared error. Using this technique, up to 60% of the lost capacity due to beamsteering both transmitter and receiver can be recovered.

Related Work

The capacity of wireless networks has been studied extensively, Gupta and Kumar's result being one of the most noted [26]. Gupta and Kumar's work has been extended for directional antennas [71, 56]. However, these works take an analytical approach, attempting to prove upper bounds without accounting for factors that make analysis difficult such as reflections.

There have also been several measurement studies of 60 GHz communication channels [65, 4]. One paper examines the effects of attenuation from reflection [44], and their data is leveraged for simulations in this work. These studies are often limited to a few (often one) simultaneous channels. This work studies the effect of a large number of simultaneous links.

There have also been numerous simulations of mmWave networks [45, 43, 50]. However, most of these simulations are concerned about delay spread and matching simulation with real environments. This work tries to be less myopic and instead look at an idealized environment using models with a higher level of abstraction. Using a more general constraint set, capacity, the role of interference, and how to mitigate interference are explored.

3.2 Simulator

The directional mesh network capacity simulator is architected to measure the total capacity across a highly parameterized space. At a high level, the simulator takes in a node geometry, randomly pairs the nodes together creating a topology, and has them communicate with a traffic distribution. Each node has one antenna array, and the array is oriented with respect to the link between the pairing of nodes. For a given simulation, the topology is fixed, and the nodes are either a transmitter or a receiver, never both and without alteration of its mode (transmitter/receiver). Several seeds are needed to be able to draw general conclusions of the capacity as a function of the number nodes as well as its geometry.

Three areas of the simulator are highlighted: simulation parameters, ray tracing, and executor. The simulation parameters are given to show the full capabilities of the simulator. The ray tracing subsection details the precise algorithm for computing the rays between individual transmitters and receivers. The executor highlights how traffic is turned into sets of active links between transmitters and receivers.

Parameters

The core parameter categories are: node geometry, environment configuration, antenna array model, antenna element model, beam coefficient algorithms, and traffic distribution.

Node Geometry

For node geometry, there are two main options: predetermined geometry and randomized geometry. A predetermined geometry is taken from a Graphviz file indicating the position of each node in a three dimensional space. A randomized geometry is parameterized by

the number of nodes where the position of a particular node is independent and identically distributed.

Environment Configuration

The environment configuration refers to the physical space that is being modeled.

The bounds of the space are always a rectangular cuboid determined by parameters, L, W, and H, corresponding to the limits of the x, y, and z dimension, respectively. The planes that bound the rectangular cuboid are, therefore: $X_0 : x = 0$, $Y_0 : y = 0$, $Z_0 : z = 0$, $X_L : x = L$, $Y_W : y = W$, and $Z_H : z = H$. In addition, there is also a parameter that affects the overall scale of the bounded box. This parameter is used to increase or decrease the density of the nodes. There are two benefits to using a scaling parameter:

- Easy comparison: for a given geometry, one may alter the density of nodes without changing the relative positions of the nodes.
- Higher floating point precision: a single multiply is needed in the path loss equation.

In addition to the bounds of the space, there are also parameters for controlling reflections. Reflective attenuation, $reflAtten$, controls the attenuation of a single reflection. The maximum number of reflections, $maxRefN$, is also parameterized.

Lastly, how capacity is calculated is also a parameter. Although, this body of work only used Shannon capacity as mentioned in Chapter 2. Capacity can also be calculated using a discrete set of data rates corresponding to 802.11ad. There is also a continuous version of 802.11ad based on a linear regression of 802.11ad.

Antenna Array Model

The antenna array model parameters consists of parameters that controls the number of antenna elements, the antenna spacing, the quantization of the phase and amplitude coefficients, the windowing function, randomizing of the antenna array orientation, and the roll of the antenna (i.e. the angle with respect to axis of rotation defined by the normal of the antenna array).

The simulator supports rectangular antenna arrays of any arbitrary number of antenna elements with a given spacing relative to λ. Quantization of phase and amplitude can be set separately. However, the default is to use double precision floating point representation. In addition, one may also set the roll of the antenna arrays to be some fixed constant or random. By default, the roll for a given transmitter and receiver pair is the same, but the rolls may also be generated independently by a parameter.

Several windowing functions are supported for antenna array geometries. These include: Hamming, Hanning, Kaiser, Gausian, Dolph-Chebyshev, Poisson, Lanczos, and Tukey. Lastly, one may control the deviation of a transmitter antenna array's orientation and its intended receiver antenna array's orientation using randomization. This feature is intended to mimic real world scenarios where antenna arrays are not perfectly aligned with each other.

Antenna Element Model

The antenna element model supports two antenna types for arrays, isotropic and cosine response. These elements may be used in a larger array configuration. In addition, a circular aperture is also supported. However, there is no support for a circular aperture antenna array.

Beam Coefficient Algorithms

There are several algorithms that are supported for determining the best set of coefficients for maximizing capacity. These algorithms include:

- Gradient Coordinate Descent
- Minimum Mean Squared Error
- Zero-forcing

The details of the minimum mean squared error algorithms appear later in this chapter. Two of the algorithms do not appear in this work, gradient coordinate descent and zero-forcing, as they performed worse than the minimum mean squared error algorithm. Gradient coordinate descent is a greedy algorithm that maximizes the link capacity by changing the array's coefficients one at a time by a fixed amount such that the link is maximized. Zero-forcing uses knowledge of the directions of interference to set those directions to zero.

In addition one may control where and how they are applied:

- Receiver-only adaptation. Line of sight beamforming transmitter.
- Transmitter-only adaptation. Line of sight beamforming receiver.
- Transmitter adaptation followed by then receiver adaptation.

Traffic Distribution

As mentioned in Chapter 2, traffic is generated using a Pareto ON/OFF distribution. The x_m for ON and OFF are parameters. The default is to have slotted traffic where all transmitters and receivers send synchronously. There is support for non-slotted traffic. However, the use of this parameter is not studied as it does not alter the results significantly based on experimentation.

Ray Tracing

In order to simulate the capacity of a node in a physical space, it is necessary to know the amount of power being received by the signal as well as the interferers. In addition, because nodes are not single element isotropic antennas, it is also necessary to know the angle at which the interference is received. These calculations can be very expensive for an

arbitrary room environment that is trying to mimic real world conditions. The simulator makes a number of simplifying assumptions in order to spend a higher percentage of its time in things that are of interest and to more easily able to generalize properties of directional wireless networks, (e.g. how networks scale for large number of nodes and very large (1000s of elements) antenna arrays).

Assumptions

The ray tracing assumptions are the same assumptions that the system model makes for the channel (see Section 2.3). However, with some repetition, they are explained here in the context of the ray tracer.

The simulator assumes that all reflections are from the bounding box, and these reflections are specular and have a constant multiplicative loss to the power independent of angle. Reflections only from the bounding box eliminates differences that might appear in the environment such as the placement of chairs or people. A constant multiplicative loss represents a statistical average loss that may occur in an environment. This assumption eliminates issues of calculating diffraction, polarity, and other properties of waves interacting with materials, saving computation time.

In addition, the simulator assumes that nodes are infinitely small and do not contribute to fading. This assumption follows from the assumption that nodes are relatively small compared to the containing environment and therefore fading caused by nodes will not be significant, and if it is significant, it will only be for a single node geometry and would not be true for more typical geometries.

Lastly, the simulator ignores the multipath of the intended signal and the phase of any power that is received. This assumption greatly simplifies what needs to be kept track of when calculating the power. In practice, it is very difficult to calculate because it is dependent on what data is being transmitted, and the signals for mmWave typically have a very wideband bandwidth on the order of GHz.

The ray tracer can be easily generalized to eliminate many of these assumptions such as multipath of the intended signal, polarity, and multiple different surfaces. However, what is presented here is only for the assumptions above.

Calculation

For every receiver antenna array, it is necessary to calculate the power received at all possible angles. For every transmitter antenna array, it is necessary to calculate all possible angles that can reach a given receiver that will have a significant effect on the result. A traditional ray tracer in graphics takes the optimization that the viewer of a scene is looking at a particular angle and has a finite resolution that can be seen. This allows calculation on a per pixel basis starting from the viewer and ending at the light source [3]. However, for the wireless simulator, a similar optimization proves difficult as it would require discretization of angles, which would be a source of quantization error that may effect adaptation algorithm

effectiveness. Ideally, the simulator should not introduce any additional quantization error beyond what is trying to be studied. Everything internally is represented as double precision floating point for this reason.

Instead, an optimization the simulator takes advantage of is that there is a maximum of six surfaces that may cause a reflection seen by the receiver. With this in mind, for every receiving antenna array that is active, the simulator calculates the virtual location of every transmitting antenna array. Similarly, the simulator does the same thing for every transmitting antenna array. A virtual location is the equivalent location of a node after a reflection has occurred.

For example, in Figure 3.1, imagine that the black node is the receiver, and the white node (with the solid outline) is the transmitter. The white nodes with a dashed outline are the effective virtual locations of the same transmitter, so if the receiver was omnidirectional, it would receive one line-of-sight path from the solid white node, and it would receive six non-line-of-sight paths from the dashed white nodes. Note that from the point of view of the black node, the distance, the number of reflections, and the angle are directly available simply by treating the virtual node as if it were a real node and taking into account the loss due to a reflection. This approach generalizes to an arbitrary number of reflections.

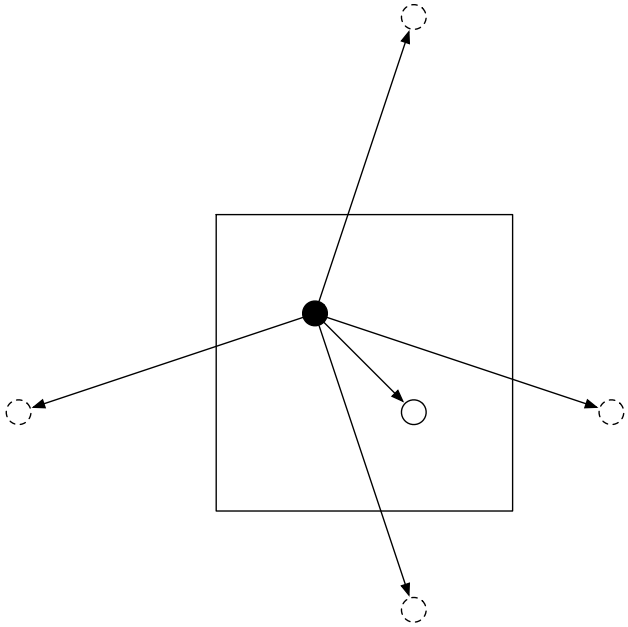
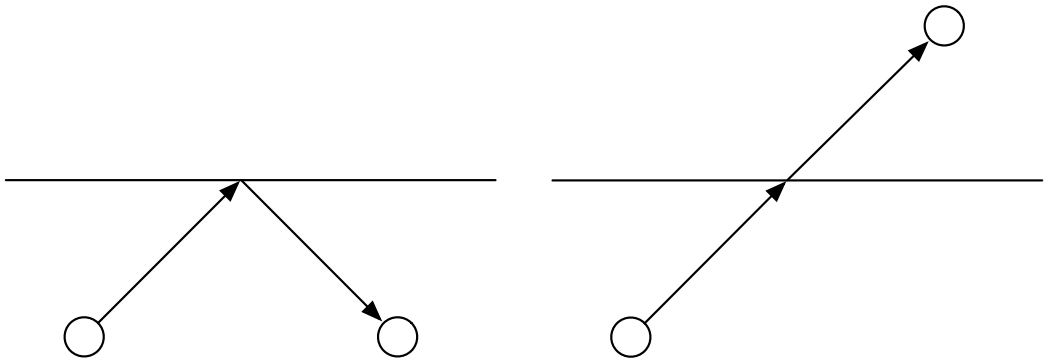


Figure 3.1: Virtual nodes (dashed outline) from the perspective of the black node transmitting to the white (solid outline) node.

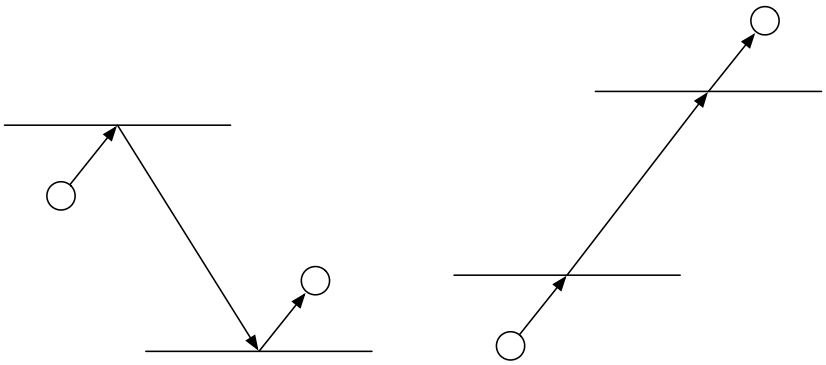
Before going into more detail as to how this is calculated, first it is good to look at a few examples. Let us consider a reflection from transmitter to receiver, Figure 3.2a, where the left node is the transmitter and the right node is the receiver. The figure shows the two rays, a ray that interfaces with the wall and the specular reflection of the wall. Imagine that



(a) Original (b) Mirrored

Figure 3.2: Single reflection from transmitter to receiver

the wall was a mirror, and you were to look at the mirror. It would appear that an identical receiver appeared on the opposite side of the wall, Figure 3.2b. Here the rays are identical in length but instead it appears that it is transmitting through the wall. Figure 3.3a shows a similar scenario except with two reflections, and Figure 3.3b shows the virtual location after the two reflections.



(a) Original (b) Mirrored

Figure 3.3: Two reflections from transmitter to receiver

Calculating a virtual location is a very simple operation requiring a single subtraction. Given a three dimensional location as $\{a, b, c\}$ and a wall w, the virtual location is calculated:

$$virtualLocation(\{a,b,c\},w) = \begin{cases} \{-a,b,c\} & w = X_0 \\ \{2L-a,b,c\} & w = X_L \\ \{a,-b,c\} & w = Y_0 \\ \{a,2W-b,c\} & w = Y_W \\ \{a,b,-c\} & w = Z_0 \\ \{a,b,2H-c\} & w = Z_H \end{cases} \tag{3.1}$$

Given a location, it is possible to generate the virtual locations of that location. However, note that for this to work recursively (e.g. generate third-order reflections from second-order reflections), it is necessary to know what wall generated the current location. Obviously, it is impossible to bounce off of the previous wall without bouncing off another wall first. The exception to this is the case where no reflections have occurred yet. Let **Walls** be the set of all walls.

$$\mathbf{Walls} = \{X_0, Y_0, Z_0, X_L, Y_W, Z_H\} \tag{3.2}$$

Every virtual location, l, consists of both the position, p, as well as the previous walls that generated it, w_p.

$$l = (p, w_p) \tag{3.3}$$

The list of virtual locations from a virtual location is given as:

$$nextVLoc((p, w_p)) = [\forall w \in \mathbf{Walls} - \{head(w_p)\}, (virtualLocation(p, w), w : w_p)] \tag{3.4}$$

Note that *head* takes the head of the list; if the list is empty, it returns empty. For the initial location, w_p is the empty list, [].

The ray tracer algorithm, $rayTrace$, is a recursive function that calls $nextVLoc$. It takes as arguments the current recursive depth n, the original location o, and a function f that is applied to every virtual location. The function f takes as arguments the reflection-order and the virtual location and may return anything in return. In practice, f is used to dynamically decide whether or not a virtual location should be kept. For example, because there is no response 'behind' an antenna array, the simulator may reject that virtual location, leading to optimizations discussed later.

The ray tracer algorithm itself is very simple. If the recursive depth is zero, return the empty list. Otherwise, call f on every virtual location and concatenate ($+\!\!+$) it with the recursive call to ray trace on the new virtual locations. Note to determine the reflection order, it is necessary to subtract the recursive depth from the $maxRefN$ and offset it by one.

$$\begin{aligned}
&rayTrace :: \forall a.(\mathbb{N} \to (\mathbb{R}^3, [\mathbf{Walls}]) \to a) \to \mathbb{N} \to (\mathbb{R}^3, [\mathbf{Walls}]) \to [a] \\
&rayTrace(f, 0, o) = [] \\
&rayTrace(f, n, o) = \\
&\qquad map(\lambda l.f(maxRefN - n + 1, l), nextVLoc(o)) +\!\!+ \\
&\qquad concat(map(\lambda l.rayTrace'(f, n - 1, l), nextVLoc(o)))
\end{aligned} \tag{3.5}$$

The ray tracer has complexity $O(A5^{maxRefN})$ where A is the number of active antenna arrays. Even though this algorithm is exponential, the exponent tends to be small for most environments of interests (typically 12 or less). For larger environments $maxRefN$ is typically very small due to path loss being the dominant source of loss in the channel. Additionally, as A increases, the $maxRefN$ decreases for a constant size volume due to the increased amount of interference from lower order reflections. For more information as to how to set $maxRefN$, please refer to Appendix B.

The ray tracer in Equation 3.5 may be easily modified to stop considering virtual locations after it has reached a certain amount of path and reflective loss, potentially reducing the complexity of the ray tracer considerably. The simulator does not make this optimization because the ray trace results for a given run are saved and reused for different size antenna arrays. For example, a path that may be below thermal noise for a small antenna array may be significantly above thermal noise for a larger antenna array. Saving the ray trace results also allows easy scaling of a particular environment without additional ray tracing (e.g. changing the node density by turning a $3\,\mathrm{m}^3$ cube into a $30\,\mathrm{m}^3$ cube).

The ray tracer algorithm alone does not distinguish if a particular virtual location is necessary or not. A necessary condition for a virtual location to be considered of interest is that the gain in the direction of the virtual location with respect to both the transmitting and receiving antenna array be greater than zero. Note that since antenna arrays have a grounded backplane a large number of virtual locations will be discarded (approximately half).

For a ray result to be valid, we must ensure that both the transmitter and receiver's paths are both valid. We can take advantage of the symmetry between the ray from transmitter to receiver and receiver to transmitter to only ray trace once instead of twice. To do this, given a list of walls and an original position, take the list and successively compute the virtual location, $reverseRay$. This is simply done with a fold left operation, $foldl$.

$$\begin{aligned} &reverseRay :: \mathbb{R}^3 \to [\mathbf{Walls}] \to (\mathbb{R}^3, \mathbf{Walls}) \\ &reverseRay(x, ws) = foldl(virtualLocation, x, ws) \end{aligned} \tag{3.6}$$

We must define a function, $joint$, for the ray tracer algorithm such that it will do both the transmitter and receiver jointly. $joint$ takes as input a function that determines if the virtual location is valid from the point of view of the receiver and any result it may choose, rx; the same function as previous except from the point of view of the transmitter, tx; the original transmitter location, $oTxP$; the number of bounces, n; and a virtual location, (p, ws). $joint$ returns a function whose return type is a tuple consisting of the results of the two functions and a boolean indicating if the ray is valid. We use the $reverseRay$ function to compute the virtual location from the point of view of the receiver by using the list of walls and the original transmitter location.

$$\begin{aligned} &joint :: \forall a.((\mathbb{R}^3, [\mathbf{Walls}]) \to (\mathbb{B}, a)) \to ((\mathbb{R}^3, [\mathbf{Walls}]) \to (\mathbb{B}, a)) \to \mathbb{R}^3 \\ &\quad \to \mathbb{N} \to (\mathbb{R}^3, [\mathbf{Walls}]) \to (\mathbb{B}, (a, a)) \\ &joint(rx, tx, oTxP, n, (p, ws)) = \\ &\quad \mathrm{let}(txB, txR) = tx(n, (reverse(ws), p)) \\ &\quad\quad (rxB, rxR) = rx(n, (ws, reverseRay(oTxP, ws))) \\ &\quad \mathrm{in}(txB \&\& rxB, (txR, rxR)) \end{aligned} \tag{3.7}$$

For convenience, we will define a function, fvo, that will filter only the valid outputs.

$$\begin{aligned} &fvo :: \forall a.[(\mathbb{B}, a)] \to [a] \\ &fvo(xs) = map(snd, filter(fst, xs)) \end{aligned} \tag{3.8}$$

The algorithm, *align*, that does the ray tracing between a transmitter and receiver is given below. It takes as input the position of the receiver, rxP, and a function that determines if the virtual location is valid as well giving any other output it desires, rxF. There is a similar input for the transmitter as well, (txP, txF).

$$\begin{aligned} &align :: \forall a.(\mathbb{R}^3, (\mathbb{R}^3, [\mathbf{Walls}]) \to (\mathbb{B}, a)) \to (\mathbb{R}^3, (\mathbb{R}^3, [\mathbf{Walls}]) \to (\mathbb{B}, a)) \to [(a, a)] \\ &align((rxP, rxF), (txP, txF), f) = \\ &\quad fvo(rayTrace(joint(rxF, txF, txP), maxRefN, (rxP, []))) \end{aligned} \tag{3.9}$$

Executor

The executor handles the actual simulation of the traffic. The executor's role is to turn a list of messages from nodes in continuous time into a list of discrete sets of links that are active simultaneously. It takes a function as input to handle how these sets of active links should be computed.

It is a high-level function that takes as input: a default return value, s; a solver function, $SolverFunc(s)$, and a list of messages, $[Message]$. The default value is polymorphic. The solver function is a function that takes as input a set of links that are active, *ActiveLinks*, and computes a result, s. Each message consists of a source, destination, start time, and a flag indicating if it is the beginning or end of the message. The list messages are ordered by start time. The executor returns the results of the simulation, $SimulationResults(s)$. $SimulationResults(s)$ is a tuple of a table whose key is every *ActiveLinks* that occurred and value is the result; and a sequence of tuples consisting of the time and result at that time.

$$executor :: s \to SolverFunc(s) \to [Message] \to SimulationResults(s) \tag{3.10}$$

For each message, links are added or removed based on the messages flag. When a link is added or removed, the new result using the solver function is computed if the next message is a different start time than the message that is being processed. Otherwise, new messages continue to be processed. The precise details of the executor are given in Appendix A.

3.3 Algorithms

Two different algorithms are explored: No-Adaptation and Receiver-Adaptation. No-Adaptation uses line of sight beamforming (i.e. beamsteering) for both transmitter and receiver. Receiver-Adaptation uses line of sight beamforming for the transmitter and minimum mean squared error (MMSE) adaptation for the receiver.

No-Adaptation

For this algorithm, the transmitter and receiver form beams in the direct line of sight path, independent of the channel state. The algorithm requires as input the direction, $\vec{v}_s$, of the

other side of the link. In a practical system, this direction is found using channel sounding. The algorithm for determining the beam coefficients of the antenna array is given by

$$\forall i \in E, w_i = e^{j\vec{v}_s \cdot \vec{a}_i} \tag{3.11}$$

Important to note, this algorithm relies only on phase control of the beam coefficients. The algorithm benefits from being very simple to implement, requiring computation proportional to the number of antenna elements, $O(|\mathbf{E}|)$.

Receiver-Adaptation

For receiver-adaptation, we will continue to have the transmitters use beamforming described in the previous subsection. However, we will use a classic minimum mean squared error algorithm for the receiver [66].

The minimum mean squared error algorithm takes into account the interference and noise present in the channel and attempts to maximize the ratio of signal to noise plus interference. The algorithm requires as input the desired signal's direction, $\vec{v}_s$; power, σ_s^2; and the signal received at each array element, $\mathbf{R}$.

Let $\mathbf{V}(\vec{d})$ be the set of antenna coefficients for the direction of $\vec{d}$:

$$\mathbf{V}(\vec{d}) = \{e^{j\vec{d}\cdot\vec{a}_1}; e^{j\vec{d}\cdot\vec{a}_2}; \ldots; e^{j\vec{d}\cdot\vec{a}_E}\} \tag{3.12}$$

For a given interference source, $k \in K$, let $i_k(t)$ and $\vec{v}_k$ to be the interference signal and direction of the signal, respectively. Let the noise source vector for each element be $\mathbf{N_0}(t)$. Then the received amplitude vector (each element corresponds to an antenna) is given as:

$$\mathbf{R}(t) = s(t)\mathbf{V}(\vec{v}_s) + \mathbf{N_0}(t) + \sum_{k=1}^{K} i_k(t)\mathbf{V}(\vec{v}_k) \tag{3.13}$$

The error for a given set of coefficients, $\mathbf{W}$, is then $\mathbf{W}^T\mathbf{R}(t) - s(t)$. We choose our objective function to minimize the squared error:

$$\min \left|\mathbf{W}^T\mathbf{R}(t) - s(t)\right|^2 \tag{3.14}$$

Taking the derivative with respect to $\mathbf{W}$, setting it equal to zero, and solving for $\mathbf{W}$, we derive the following equation:

$$\mathbf{W} = \mathbf{M}^{-1}\sigma_s^2\mathbf{V}(\vec{v}_s) \tag{3.15}$$

where $\mathbf{M}$ is $\mathbf{R}\mathbf{R}^{\mathrm{H}}$. Noise and interference is assumed to be uncorrelated with the signal. For simulation purposes, since only power is considered, we use the following equivalent form for $\mathbf{M}$:

$$\mathbf{M} = \sigma_{noise}^2\mathbf{I} + \sigma_s^2\mathbf{V}(\vec{v}_s)\mathbf{V}(\vec{v}_s)^{\mathrm{H}} + \sum_{k=1}^{K} \sigma_k^2\mathbf{V}(\vec{v}_k)\mathbf{V}(\vec{v}_k)^{\mathrm{H}} \tag{3.16}$$

where σ_s^2 is the signal power, σ_k^2 is the interference power for source k, and $\mathbf{I}$ is the identity matrix.

The algorithm has the same complexity as a matrix inversion, which at this point in time is $O(|\mathbf{E}|^{2.373})$ [68]. For large values of $|\mathbf{E}|$, it may be more practical to use a Linear Mean Square (LMS) estimation algorithm instead [66]. However, we do not consider this algorithm as MMSE is always at least as good as LMS.

3.4 Experimental Setup

The experiments were run using the simulator described early.

To generate sets of active pairs, node communication is simulated using power delivery in slots using a Pareto On/Off distribution to mimic bursty traffic, see Section 2.4. For all experiments, OFF = 100, and ON = 10 are fixed. Slots are time synchronized across all nodes. There is no medium access control or network routing. The experiment duration is determined based on when the ratio between the Interference Free and Receiver-Adaptation has converged within a 99 percentile for all experiments.

We consider two types of node placements: random geometry and fixed geometry. In the random geometry, node placements are i.i.d. in a cubic space. An example of a random geometry including the random connections with 50 nodes is shown in Fig. 3.4. 27 experiments are run with both random geometry and topology for each parameter set. In the fixed geometry, node placements are fixed and only the connections between the nodes are randomized. The fixed geometry is derived from Berkeley Wireless Research Center where there is a node for every desk and for every cluster of nodes there is a node on the ceiling, Fig. 3.5. 27 experiments are run with different topologies to establish an average for the fixed geometry case.

The environment that is simulated takes into account path loss as well as specular reflection to determine interference power. Reflections are only considered along the six walls of the bounding box. Reflections have a constant attenuation loss independent of angle representing a statistical average loss. Reflections are simulated until the capacity converges within a 99 percentile.

Table 3.1 lists the constants for all experiments that are not varied. The constants correspond to the ones mentioned in Section 2.3. The reflection attenuation was chosen based on measurements done in [44]. The remaining constants were chosen to correspond to the 802.11ad standard [30].

We consider square antenna arrays with $\lambda/2$ spacing. Square antenna arrays are considered because node placement is i.i.d. in a three dimensional cubic space for the random geometries. If the node placement had a preferred axis such as in a convention hall where most nodes are either on the ceiling or at human level (mostly two dimensional), then one would want to increase the antenna count in the axis parallel to the floor and reduce the antenna count for the axis parallel to the walls assuming a fixed number of antennas. We will consider the case of an office space, but we will not change the antenna array dimensions to

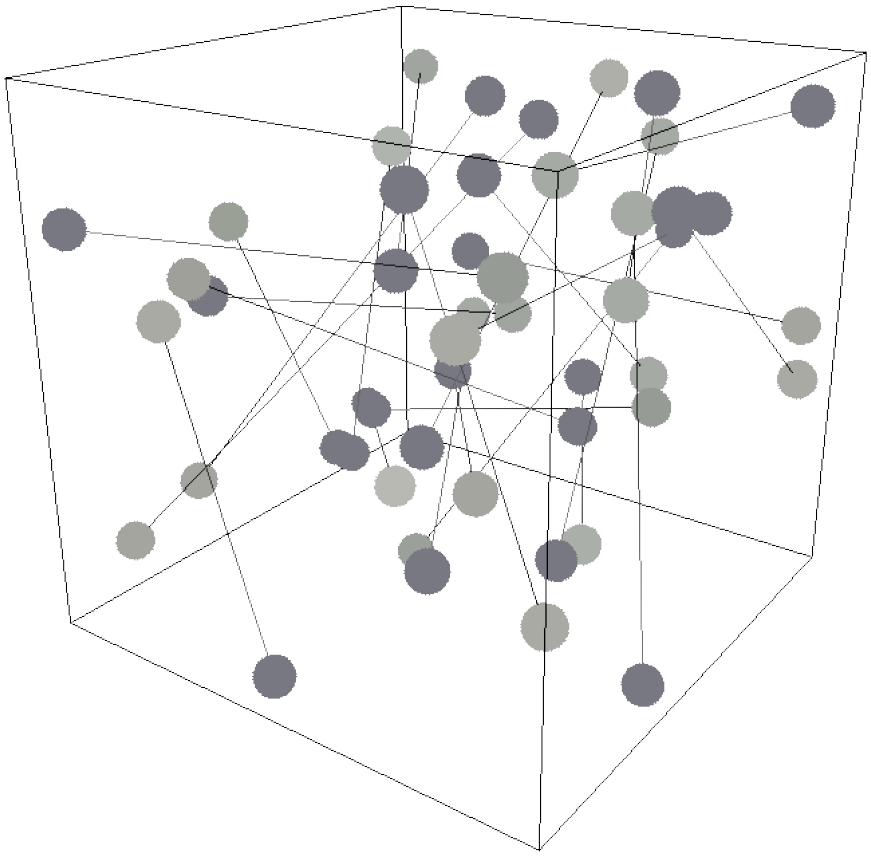


Figure 3.4: Random Geometry of 50 Nodes: Dark nodes are transmitters. Lighter nodes are receivers. Edges are connections (not beams).

be consistent with the random geometry experiments. In all of our experiments, the antenna arrays are homogeneous, no mixed antenna array dimensions.

In this study, we vary both the number of nodes as well as the square dimension of the antenna array, Table 3.2. The minimum number of nodes is set to 4 as this is the minimum configuration such that interference can occur. Transmit power is a dependent variable on the number of antenna elements to conform to an effective isotropic radiated power (EIRP) limit of 10 W based on FCC limits [16]. The fixed geometry case has a fixed number of 126 nodes. The square dimension of one is used a reference point for the pseudo-omni case. Pseudo because the antenna response pattern is a hemisphere rather than a sphere.

For all simulations, real numbers are represented using double precision floating point. Complex numbers are represented as a tuple of real and imaginary numbers, both double precision floating point.

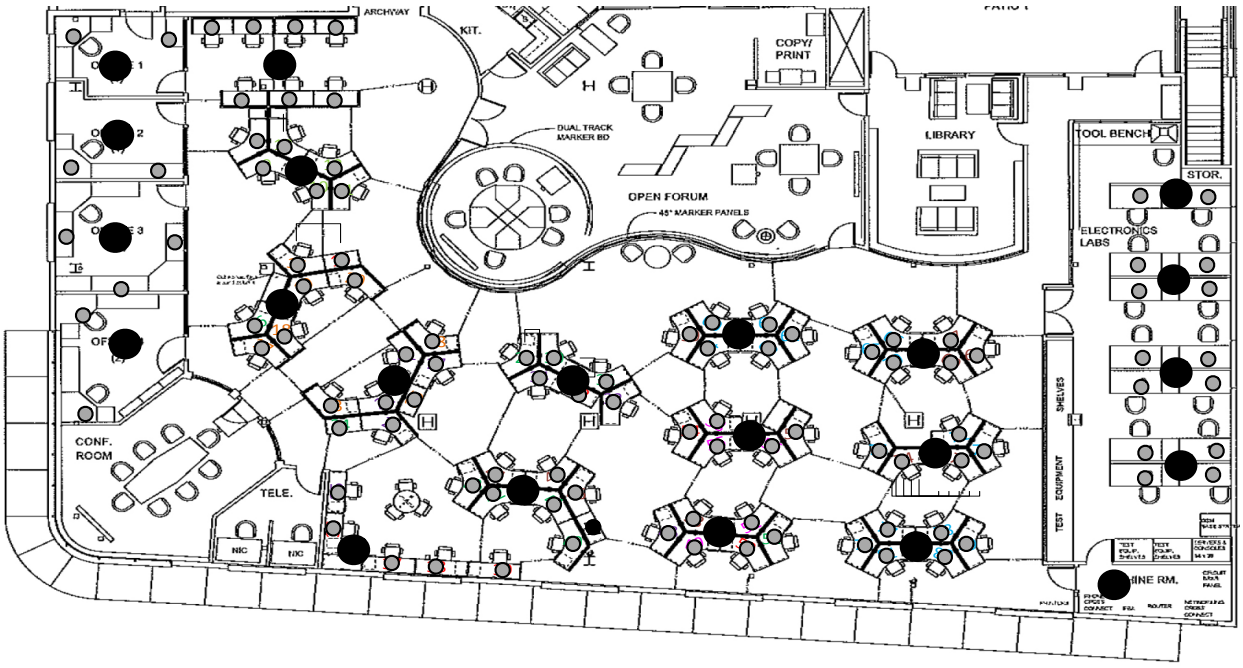


Figure 3.5: Fixed Geometry: Berkeley Wireless Research Center. Grey nodes are desk nodes, elevation 1.5 m. Black nodes are ceiling nodes, elevation 3.3 m.

Table 3.1: Simulation Constants

Parameter Name	Constant	Value
Carrier Wavelength	λ	5 mm
Bandwidth	bw	2.16 GHz
Reflection Atten	$reflAtten$	10 dB
Noise Figure	$noiseFig$	10 dB
Implementation Loss	$implLoss$	5 dB
Room Temperature	$temp$	290 K
Random Geom Dim	$H \times L \times D$	3 m×3 m×3 m
Fixed Geom Dim	$H \times L \times D$	4.5 m×25.5 m×15 m

3.5 Results

For both the random and the fixed geometry, the two algorithms, no-adaptation and receiver-adaptation, perform relatively worse than the interference free case. In other words, completely eliminating interference is not possible for either algorithm. Unsurprisingly, if one considers only the line of sight interference, most of the interference comes from sidelobes from either transmitter or receiver for large antenna arrays. Mainlobe to mainlobe interference, for large arrays, is rare as one might expect (next subsection).

Table 3.2: Simulation Variables

Parameter Name	Variable	Values
Number of Nodes	N	4,10,20,30,40,50
Square Dimension	$\sqrt{\lvert\mathbf{E}\rvert}$	1,2,4,8,16,32
Transmit Power	txP	$10/\lvert\mathbf{E}\rvert$ W

However, reflections play a very large role in interference. In simulation of the no-adaptation case, approximately two reflections for both geometries are necessary to accurately account for interference. Receiver-adaptation requires twelve and eight reflections for random and fixed geometries, respectively, due to MMSE taking advantage of the degrees of freedom of the antenna array. If the number of directions of unique interference sources is less than the total number of antenna elements minus one, one can almost perfectly cancel out the interference. However, given enough reflections, there is never a case where there are enough degrees of freedom to cancel out all interference for a practically sized antenna array.

As an aside, from our experiments, we have determined that altering the EIRP does have an effect on interference. However, changing the power by an order of magnitude results in less than a 10% change in relative capacity to interference free.

The results in Fig. 3.7,3.8,3.9 show the distribution of points for each random seed. The lines are drawn through the average of the set of points. The points are all the relative capacity compared to interference free.

Effect of Sidelobes

The effect of sidelobes were simulated by running several scenarios with and without sidelobes. To simulate a beam without sidelobes, the 3 dB mainlobe beam width was taken and anything that was not in the mainlobe was considered to be zero. These experiments were compared against those where the full antenna response was taken into account. The result for all of these scenarios was that if there was a sufficient number of antenna elements, all of the interference could be removed from the system, maximizing capacity, simply by using line of sight beamforming for both transmit and receive. An example result for one of these scenarios is shown in Figure 3.6.

Random Geometry

The performance for all sized antenna arrays as a function of the number of nodes is similar, Fig. 3.7. As one would expect, as the number of nodes is increased, the impact of interference on capacity increases. For smaller arrays, we see that the impact of interference is very high. For example, for 50 nodes and a 4×4 array, we see that on average almost half of the capacity is lost because of interference when we do not apply any adaptation. For the

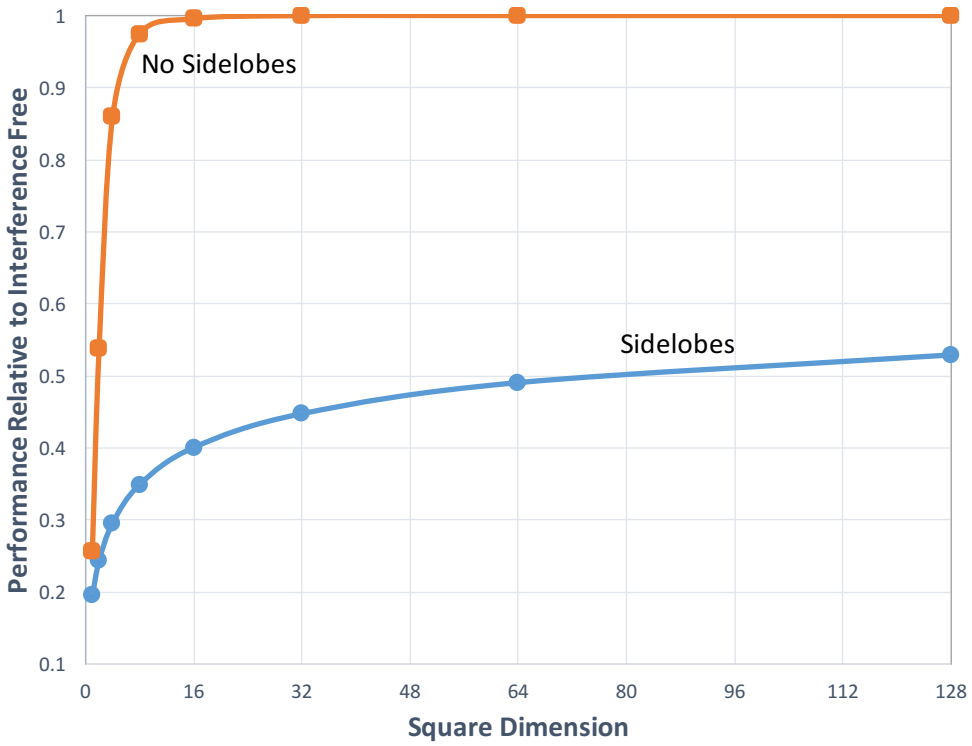


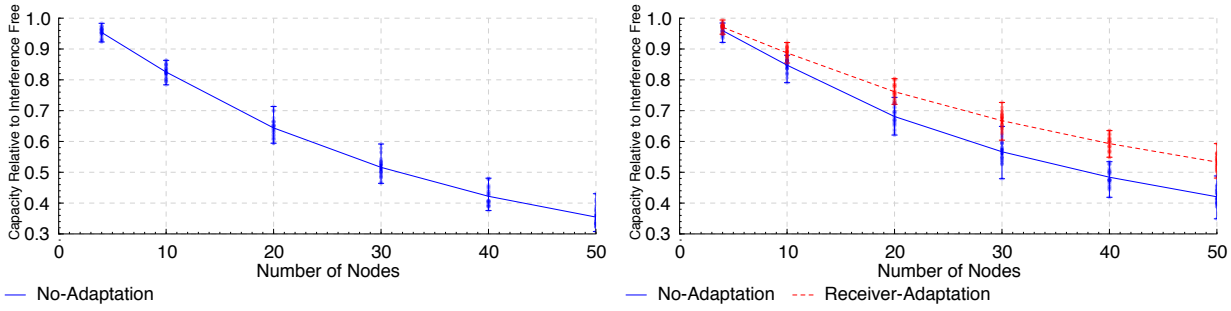
Figure 3.6: The Effect of Sidelobes on Interference

receiver-adaptation, we are able to reduce the amount of capacity lost to interference to 70%. This is a 30% improvement over the no-adaptation. Looking at larger arrays (Fig. 3.7e and 3.7c), we see that there is an improvement of at least 50% if not higher across all node densities (up to 60%).

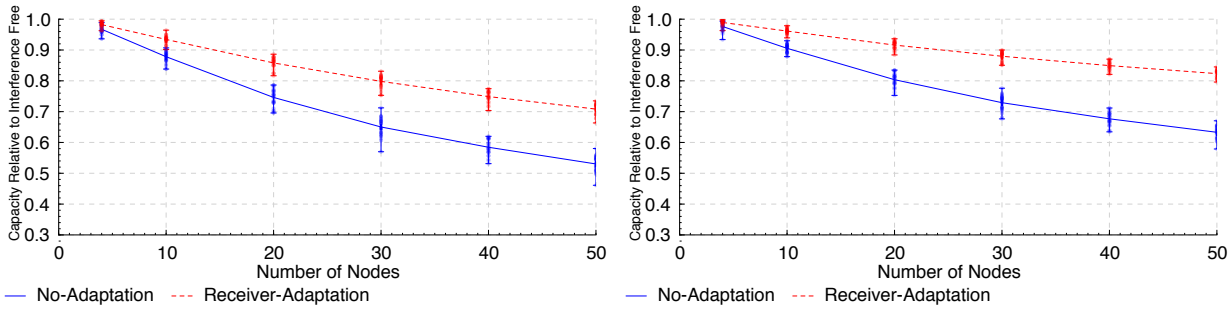
Figure 3.8 shows the same data except plotted against the antenna array size. There is no pseudo-omni data point for receiver adaptation because there is no adaptation possible with a single antenna. As one might expect, larger antenna arrays lead to less impact on capacity by interference. Here again, we can see that for large arrays we realize at least a 50% improvement in capacity for larger arrays. We see that the impact of receiver-adaptation is highest when moving from a 2×2 array to a 4×4 array for all node densities.

Fixed Geometry

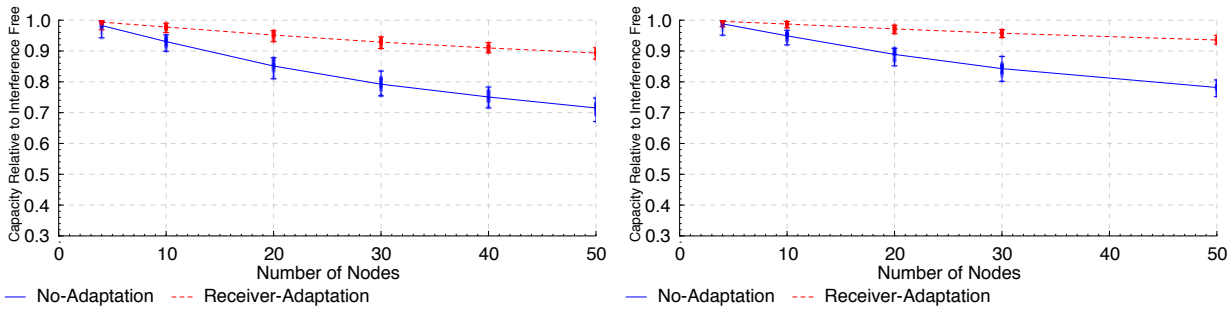
For the fixed geometry case, we see a similar scaling with antenna array size as the random geometry case. Interestingly, if we compare the pseudo-omni case against a 2×2 array, we see that the impact of interference for no-adaptation actually increases. This is because the beam for such a small array is rather large, so we are in fact receiving an increased relative gain from interference power. However, the total capacity does continue to increase with antenna count.



(a) Single Element Antenna



(b) 4×4 Antenna Array



(c) 16×16 Antenna Array

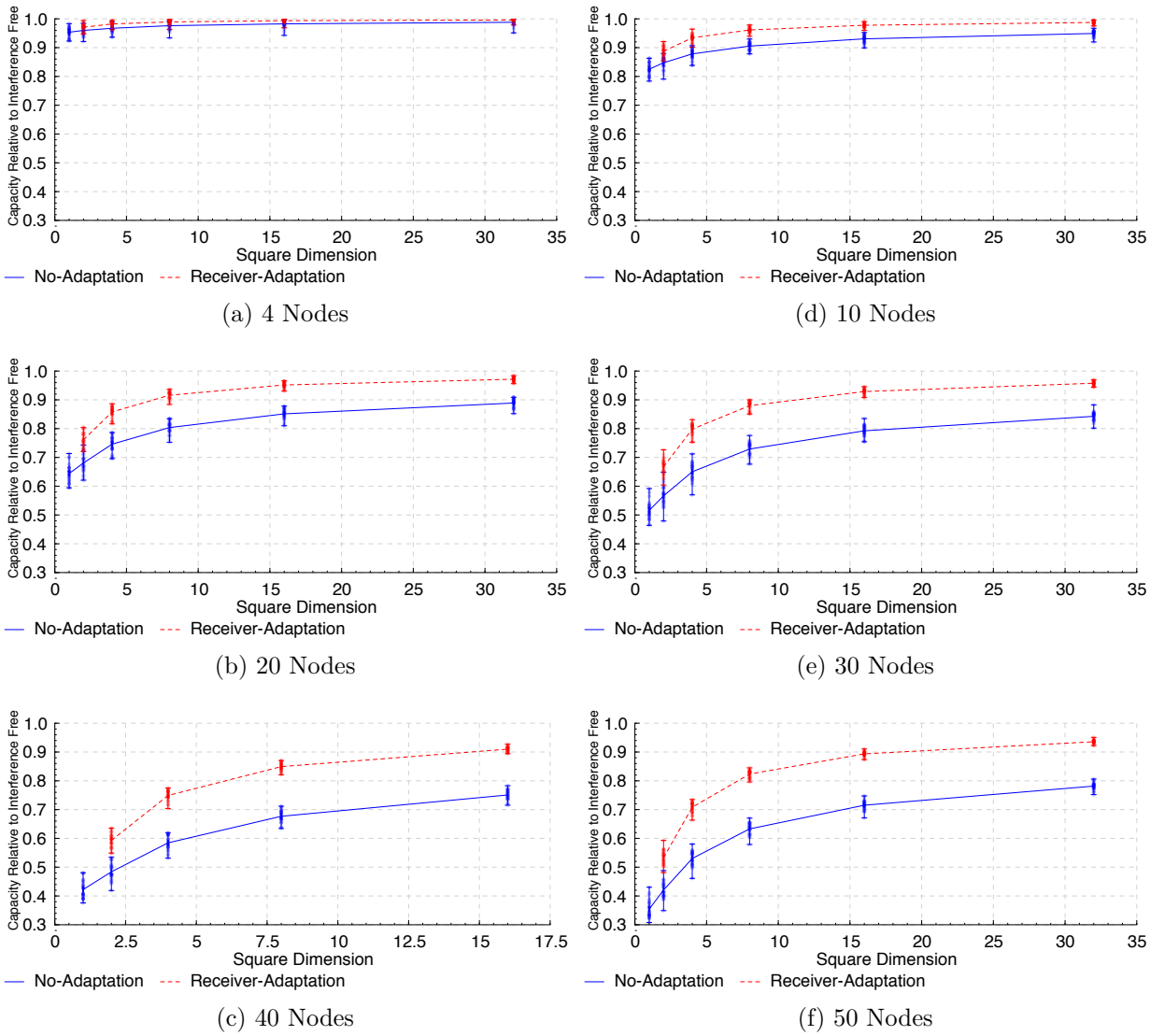
(d) 2×2 Antenna Array

(e) 8×8 Antenna Array

(f) 32×32 Antenna Array

Figure 3.7: Random Geometry: Relative capacity as a function of number of nodes compared to interference free.

For this extreme case of 126 nodes, we are still able to recover up to 42% of the capacity that was lost due to interference using receiver-adaptation for arrays 8×8 and up. As mentioned earlier, we did not optimize the antenna array dimensions. Given the geometry of the space, it would make sense to change the dimensions, and better capacity recovery should be realizable.



(a) 4 Nodes

(d) 10 Nodes

(b) 20 Nodes

(e) 30 Nodes

(c) 40 Nodes

(f) 50 Nodes

Figure 3.8: Random Geometry: Relative capacity as a function of antenna array size compared to interference free.

3.6 Summary

From these experiments, I have shown interference has a large impact on network capacity for even very high antenna count arrays. In particular, we cannot treat the beams from large arrays as being "pseudo-wires." An accurate model of sidelobes is critical! I have shown that using line of sight transmitter beamforming and receiver adaptive beamforming we can recover approximately half of the capacity lost due to interference.

Further work would explore the effects of non-line of sight paths when line of sight is not available. In addition, the effect of different antenna array geometries and quantization

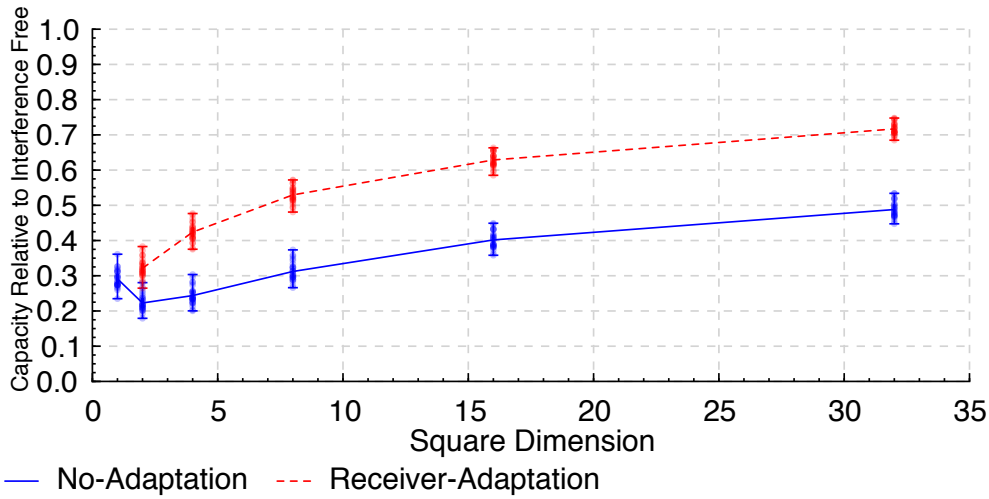


Figure 3.9: Fixed Geometry: Relative capacity as a function of antenna array size.

of antenna coefficients should also be studied. While this study is promising, for a given scenario, I am unable to compare what the true upper bound of capacity is without an exhaustive, exponential search. How much more improvement is possible if we also adapt the transmitter's beams as well?

Chapter 4

Antenna Array Geometries

There is an open problem as to what type of antenna array geometry is optimal for a given scenario. In this chapter, the problem is addressed from the point of view of interference with other nodes. Antenna array geometries are examined in directional wireless networks and their effect on interference using probabilistic analysis. Nodes are treated as having a uniform distribution in a given physical space, and the expected intereference is calculated. Linear antenna arrays, independent of position, perform significantly better than other rectangular antenna array geometries given a fixed number of antennas.

4.1 Introduction

An interesting property of antenna arrays is that the maximum gain of an antenna array is determined by the number of antenna elements and is independent of antenna geometry if the distance between antenna elements is relatively small compared to the distance between different antenna arrays. Changing the antenna array geometry has only an effect on the beam pattern shape. Therefore, one may conclude that antenna array geometry for a fixed number of antenna elements should not have a significant effect on capacity if there is no pattern to the interference. It was thought, later proved wrong, that this lack of an effect was true for a significant number of nodes placed i.i.d. in a finite volume. However, the fallacy in this argument is that a boundary condition happens due to nodes communicating with other nodes in the same finite volume.

Through experiments in interference mitigation, it was determined that the antenna array geometry had an appreciable effect on capacity. Particularly, the square antenna array had a smaller system capacity compared to other antenna arrays even though the position of nodes were i.i.d. in a cube. Therefore, this observation inspired an approach to investigate the effect of antenna array geometry using analytical tools abstracting away network properties such as the number of nodes, topology, transmission rates, and path loss.

Nodes' orientation are determined by a probability density function that is based on a node needing to communicate to other nodes. This function is used to calculate the expected

gain in a particular direction using integration. The expected gain is calculated for different antenna array geometries and compared. Because this is a comparison study of antenna array geometries, parameters such as path loss may be ignored. The parameters that have an effect on the expected gain are the antenna array geometry and the volume under consideration.

Related Work

The most closely related work was done by Peter Bevelacqua as part of his dissertation [8]. He uses optimization techniques to optimize the antenna array geometry given an interference pattern. His work takes into account antenna mutual coupling, which is not considered, as modern engineering techniques can minimize this effect such as [1].

His approach does not take into consideration the boundary effects of the position of nodes. In addition, his work examined only a few antenna elements, a maximum of seven antennas. Therefore, his results prove to be very difficult to compare against.

4.2 Probability Density Function

For a given node position, it is needed to define the likelihood of a given array orientation. A reasonable assumption would be that an antenna array is more likely to be oriented in a direction that has a higher chance of having another node in that direction. Therefore, in a finite space, this likelihood is proportional to the distance of the array to the surface it is facing.

Contributions from reflections are ignored because they will always be of lower gain because of reflective loss and increased path length and therefore would not be preferred. In addition, the boundary effect is present independent if the boundaries are reflective or simply there are no nodes present beyond the boundaries. The only case where it might be important to model is in the case of occlusion without relays necessitating the need for reflections, which is ignored.

Given a position, the likelihood of a given direction is defined to be the length of the ray to the surface it interacts with inside the area or volume (depending on the number of dimensions). To create a probability density function, the above function is normalized by integrating over all possible directions for a given position.

In this work, rectangular cuboids are considered of size $L \times W \times H$, which correspond to the x-, y-, and z-axis, respectively. In two dimensions, the z-axis is disregarded. Spheres are also examined as a point of comparison where there is only one wall.

Two Dimensions

To derive an expression for the probability distribution function, given a position $p = (p_x, p_y, p_z)$, where $p_z = 0$ in 2D, and an angle of direction ϕ, the face at which an array is pointing needs to be determined. Figure 4.1 gives a reference for the symbols use in

relation to the XY-plane. There are four unique angles that determine the face. Using

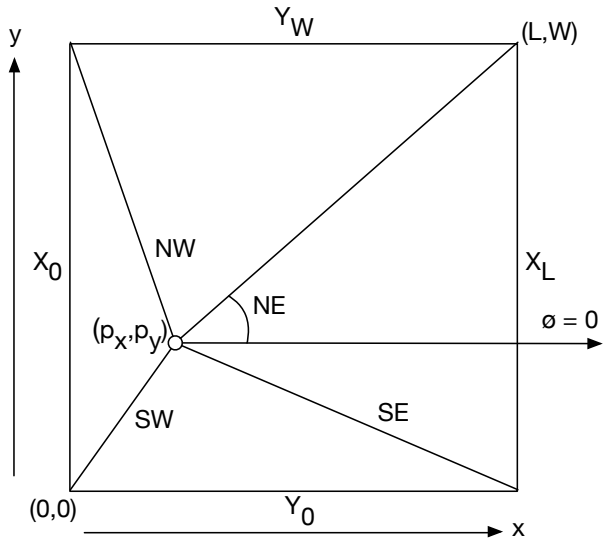


Figure 4.1: XY-plane

trigonometry, the angles are determined that define the faces.

$$NE(p) = \arctan \frac{W - p_y}{L - p_x} \tag{4.1a}$$

$$NW(p) = \arctan \frac{p_x}{W - p_y} + \frac{\pi}{2} \tag{4.1b}$$

$$SW(p) = \arctan \frac{p_y}{p_x} + \pi \tag{4.1c}$$

$$SE(p) = \arctan \frac{L - p_x}{p_y} + \frac{3\pi}{2} \tag{4.1d}$$

The faces are labeled X_0, X_L, Y_0, Y_W where X_0 refers to the $x = 0$ plane, Y_0 refers to the $y = 0$ plane, and etc.

$$face_{2D}(p, \phi) = \begin{cases} Y_W & NE(p) < \phi \leq NW(p) \\ X_0 & NW(p) < \phi \leq SW(p) \\ Y_0 & SW(p) < \phi \leq SE(p) \\ X_L & \text{otherwise} \end{cases} \tag{4.2}$$

Knowing the face f, the length of the ray, l_{xy}, can be determined using trigonometry as

well.

$$l_{xy}(p,\phi,f) = \begin{cases} \frac{W-p_y}{\sin\phi} & f = Y_W \\ \frac{-p_x}{\cos\phi} & f = X_0 \\ \frac{-p_y}{\sin\phi} & f = Y_0 \\ \frac{L-p_x}{\cos\phi} & f = X_L \end{cases} \tag{4.3}$$

The two-dimensional probability density function of ϕ is the length of the ray at ϕ normalized by the integral over all ϕ for the given position p.

$$pdf_{2D}(p,\phi) = \frac{l_{xy}(p,\phi,face_{2D}(p,\phi))}{\int_0^{2\pi} l_{xy}(p,\phi',face_{2D}(p,\phi'))d\phi'} \tag{4.4}$$

Obviously, from the above PDF, one can expect that the expected orientation of an antenna array geometry is the vector that goes through the center. The farther away an antenna array is from the center, the more biased an array is toward pointing the center, Fig. 4.2. We will not further consider the two-dimensional case, as one can only build linear arrays in two dimensions, assuming planar arrays.

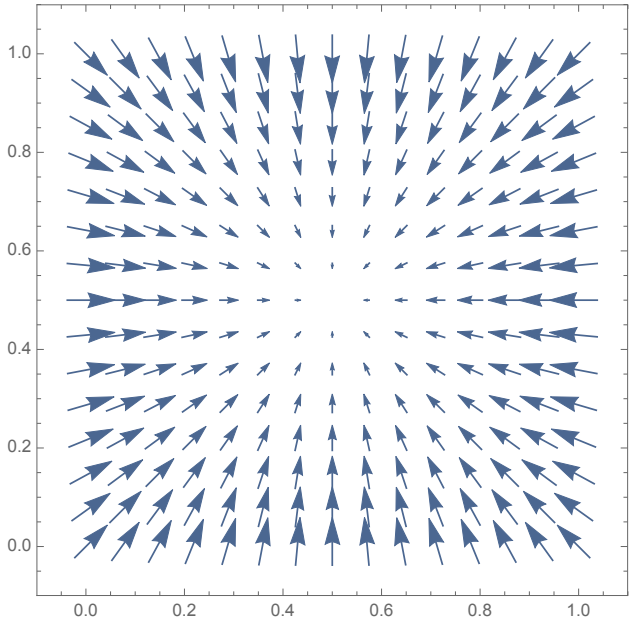


Figure 4.2: Expected orientation of an antenna array in 2D

Three Dimensions

For three dimensions, we need to know which of the six faces the ray terminates at. We use spherical coordinates as defined by ISO, and let the unit vector at $\theta = 0$ be equivalent to the vector $(0, 0, 1)$. Figure 4.3 gives the reference for symbols in the z-axis.

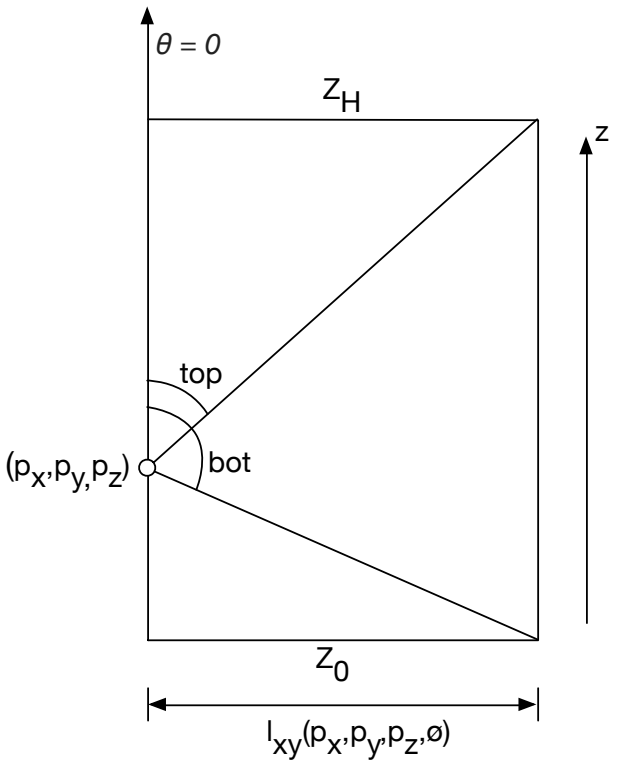


Figure 4.3: Z-axis

The angles where the ray transitions to either being the top or bottom of the rectangular cuboid is determined by the length of the ray in the xy-plane in addition to the elevation angle, θ.

$$top(p, \phi, f) = \arctan \frac{l_{xy}(p, \phi, f)}{H - p_z} \tag{4.5a}$$

$$bot(p, \phi, f) = \arctan \frac{p_z}{l_{xy}(p, \phi, f)} + \frac{\pi}{2} \tag{4.5b}$$

If θ is above the *top* angle, then it hits with the Z_H plane. If θ is below the *bot* angle, the it hits the Z_0 plane. Otherwise, it is hitting a face as defined by the two-dimensional $face_{2D}$ function.

$$face'_{3D}(p, \phi, \theta, f) = \begin{cases} Z_H & \theta \leq top(p, \phi, f) \\ Z_0 & \theta > bot(p, \phi, f) \\ f & \text{otherwise} \end{cases} \tag{4.6a}$$

$$face_{3D}(p, \phi, \theta) = face'_{3D}(p, \phi, \theta, face_{2D}(p, \phi)) \tag{4.6b}$$

The length of the ray in 3D, l_{xyz}, is determined by using trigonometry.

$$l_{xyz}(p,\phi,\theta,f) = \begin{cases} \frac{-p_z}{\cos\theta} & f = Z_0 \\ \frac{H-p_z}{\cos\theta} & f = Z_H \\ \frac{-p_x}{\sin\theta\cos\phi} & f = X_0 \\ \frac{W-p_y}{\sin\theta\sin\phi} & f = Y_W \\ \frac{-p_y}{\sin\theta\sin\phi} & f = Y_0 \\ \frac{L-p_x}{\sin\theta\cos\phi} & f = X_L \end{cases} \tag{4.7}$$

We must normalize the probability density function for a given position. We do this by integrating over all spherical coordinates.

$$norm(p) = \int_0^{2\pi}\int_0^{\pi} l_{xyz}(p,\phi,\theta,face_{3D}(p,\phi,\theta))\sin\theta d\theta d\phi \tag{4.8}$$

The probability density function in three dimensions is the length of the ray normalized.

$$pdf(p,\phi,\theta) = \frac{l_{xyz}(p,\phi,\theta,face_{3D}(p,\phi,\theta))}{norm(p)} \tag{4.9}$$

Sphere

The sphere's probability density function is more simply defined as there is only one surface. Given a direction specified in spherical coordinates, the point on the surface of the sphere, $p_{\bigcirc}$, that corresponds to it is given by a standard spherical coordinate definition according to ISO standards.

$$p_{\bigcirc}(\theta,\phi) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) \tag{4.10}$$

Therefore, the distance between a given point, p, and the surface sphere, is the magnitude of the difference of the two points.

$$l_{\bigcirc}(p,\theta,\phi) = ||p - p_{\bigcirc}(\theta,\phi)|| \tag{4.11}$$

Normalizing by integrating over all spherical coordinates. The probability density function of a sphere is given as:

$$pdf_{\bigcirc}(p,\theta,\phi) = \frac{l_{\bigcirc}(p,\theta,\phi)}{\int_0^{2\pi}\int_0^{\pi} l_{\bigcirc}(p,\theta',\phi')\sin\theta' d\theta' d\phi'} \tag{4.12}$$

4.3 Expectation

Given the probability density function defined above, it is of interest to know what the expected performance of an antenna array geometry would be relative to other antenna array geometries.

The expectation of a point given a value function, v, is given as the value function multiplied by the probability density function of the node across all directions multiplied by the probability density function across all nodes and their directions, Eq. 4.13. Note that we assume that each node position is independent. Also note that the probability density functions are configured such that the directions can be treated as independent.

$$\begin{aligned} EGain(p) = \int_0^L \int_0^W \int_0^H \int_0^{2\pi} \int_0^{\pi} \int_0^{2\pi} \int_0^{\pi} v(p, \theta, \phi, (p'_x, p'_y, p'_z), \theta', \phi') \cdot \\ pdf(p, \phi, \theta) pdf(p'_x, p'_y, p'_z, \phi', \theta') \sin(\theta) \sin(\theta') d\theta' d\phi' d\theta d\phi dp'_z dp'_y dp'_x \end{aligned} \tag{4.13}$$

In our study, we assume that the antenna array's steering direction is in the same direction to the antenna array's orientation. We assume this for two reasons: the steering direction follows the same probability density function as the antenna array's orientation; and the antenna pattern itself is not significantly different except close to endfire, which should be rare. Following this assumption, the antenna array coefficients are all one, $\mathbf{W} = \mathbf{1} = \{1; 1; \ldots 1\}$.

The value function is defined as being the multiplicative gain of the two arrays. In the equation below, (ϕ, θ) correspond to the orientation of the antenna array at position p. $angle$ is a function that computes the relative angle with respect to the antenna array. This may be computed with any vector rotation algorithm such as Rodrigues' rotation formula [51]. The roll of the antenna array (i.e. the axis of rotation defined by the normal of the planar array) is randomly varied.

$$\begin{aligned} &v(p, \phi, \theta, p', \phi', \theta') = \\ &\quad gain(angle(p' - p, \phi, \theta), \mathbf{1}) gain(angle(p - p', \phi', \theta'), \mathbf{1}) \end{aligned} \tag{4.14}$$

Note that the value function does not include path loss. We do not include it because we are concerned with the relative performance of the antenna arrays. In this case, the path loss would be the same for all antenna arrays. This has the nice side effect that we need only be concerned with the ratio of the space we are considering and can ignore absolute dimensions.

4.4 Volume of Linear Array vs Square Array

An interesting property that arises from the probability density function described is that for nearly all probable directions the volume of a linear array is smaller than other rectangular antenna arrays with the same number of elements. This means that the linear array is less likely to cause interference for the same antenna cost. We will examine more closely this

effect for the mainlobe as it has the highest gain and take the square antenna array as an example.

Given a wavelength spacing between elements, s, the equation that describes the 3 dB beam angle for the mainlobe for an array as a function of number of antenna elements, n, is

$$beamAngle(n) = \left| \arccos \frac{1}{s \cdot n} - \frac{\pi}{2} \right| \tag{4.15}$$

Let us examine the case that a linear and square array project onto a boundary orthogonal to the plane. We will further assume that it will not project onto multiple boundaries, a safe assumption for narrow enough beams. There exists a minimum distance away such that

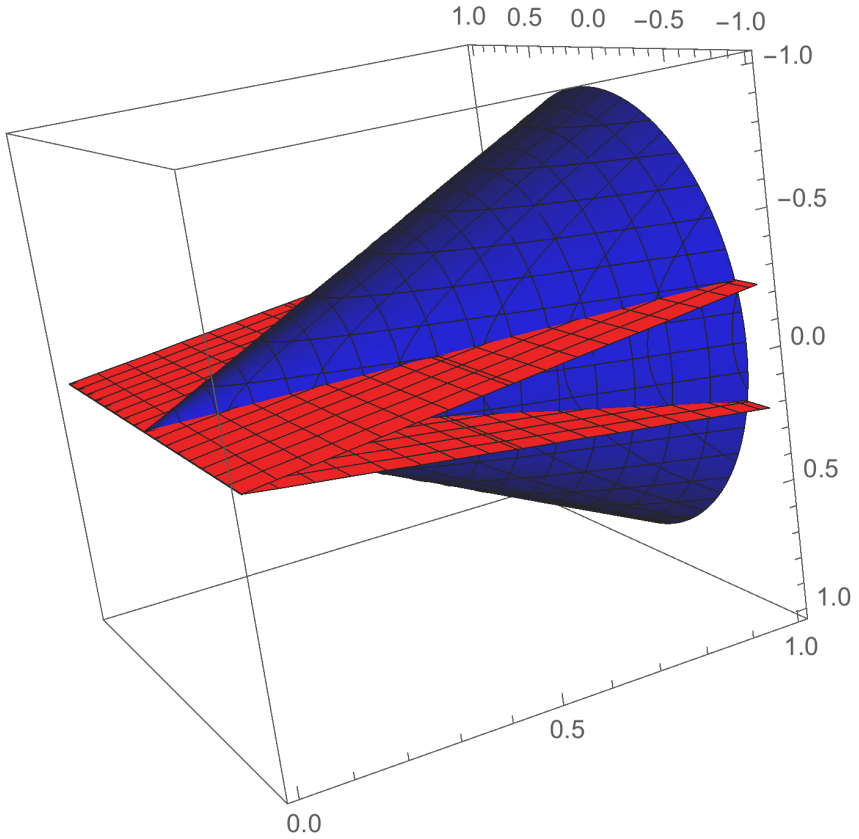


Figure 4.4: Linear (red/light) and Square (blue/dark) Mainlobe Pattern, 9 elements

the linear array's mainlobe always encompasses a smaller volume than the square array's mainlobe, *dist*, Fig 4.5.

Using the equation for conic volume for the square array's mainlobe, $volS_M$, and the equation for the uniform triangular prism volume, $volL_M$, we can determine this critical distance when the two volumes are equal. The volume of the linear array's mainlobe is bounded by the two orthogonal boundaries. The distance between these two boundaries is w. The limit converges within 90% after 50 elements.

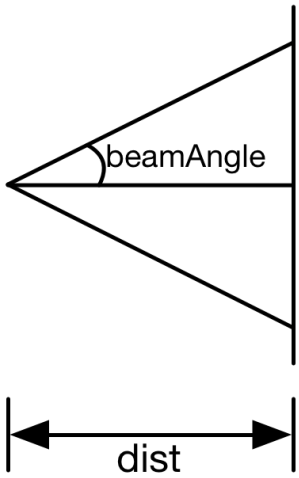


Figure 4.5: Projection of Beam onto a Boundary.

$$
\begin{aligned}
volS_M &= \frac{\pi}{3} dist^3 \tan^2(beamAngle(\sqrt{n})) \\
volL_M &= w \tan(beamAngle(n)) dist^2 \\
volS_M &= volL_M \Rightarrow dist(n) = \frac{3w \tan(beamAngle(n))}{\pi \tan^2(beamAngle(\sqrt{n}))} \\
&\text{let } s = 0.5, \lim_{n\to\infty} dist(n) = 0.477465w
\end{aligned}
\tag{4.16}
$$

From the above, if we assume lambda over two spacing, ($s = 0.5$), the square array's mainlobe is larger than the linear array's mainlobe when a node is pointed at least 47.7% of the width, w, away from the boundary that it is projecting upon. If we examine this in the context of our probability distribution function, we realize that this case is more probable than otherwise.

First consider a cube. Without loss of generality, let w be equal to 1. In a cube, nodes are more likely to point away from the nearest wall to the node. Consequently, antenna arrays are more likely to point away distances that are at least 0.5, which is greater than the minimum needed. Only those nodes close to the middle may have a smaller square array mainlobe volume than the linear array mainlobe volume, though not as likely.

Now consider the case where one of the dimensions is significantly longer than the other two dimensions. This scenario would be common, for example, in a hallway. In this scenario, nodes are more likely to point in the dimension that is longer. In that situation, w would likely be one of the smaller dimensions and therefore the mainlobe is more likely to travel greater than the minimum necessary.

The last scenario that needs to be considered is when one dimension is significantly smaller than the other two dimensions. This scenario would be common, for example, in an

exhibition hall space or terrestrial communication over large distances. In this scenario, we enter a scenario similar to the cube. A node is unlikely to point in the smaller dimension, and therefore, we have two cases to consider. If the linear array is oriented orthogonal to the smaller dimension, then w is very small relative to the other dimensions. In any other orientation, it is equivalent to the cube case.

For all of these cases, it should be noted that the relationship between the square antenna array's volume, $volS$, and the linear array's volume, $volL$, can be represented by a constant for sufficiently large number of antennas for a given antenna array position and beam direction. This includes sidelobes as they have the same shape as the mainlobe albeit with less gain. To show this, we should note from Eq. 4.16:

$$\begin{aligned} volS(n) &\propto \tan^2(beamAngle(\sqrt{n})) \\ volL(n) &\propto \tan(beamAngle(n)) \end{aligned} \tag{4.17}$$

Therefore, given a constant antenna array position and angle, there exists a C for a sufficiently large number of antenna elements, n, because in the limit the ratio of volumes between the two arrays is 2 times some constant c.

$$\lim_{n\to\infty} \frac{volS(n)}{volL(n)} = 2c = C \tag{4.18}$$

Similar equations can be derived for other rectangular arrays.

4.5 Monte Carlo Simulation

Due to the difficulty of symbolically integrating Eq. 4.13, we use numerical integration by Monte Carlo simulations.

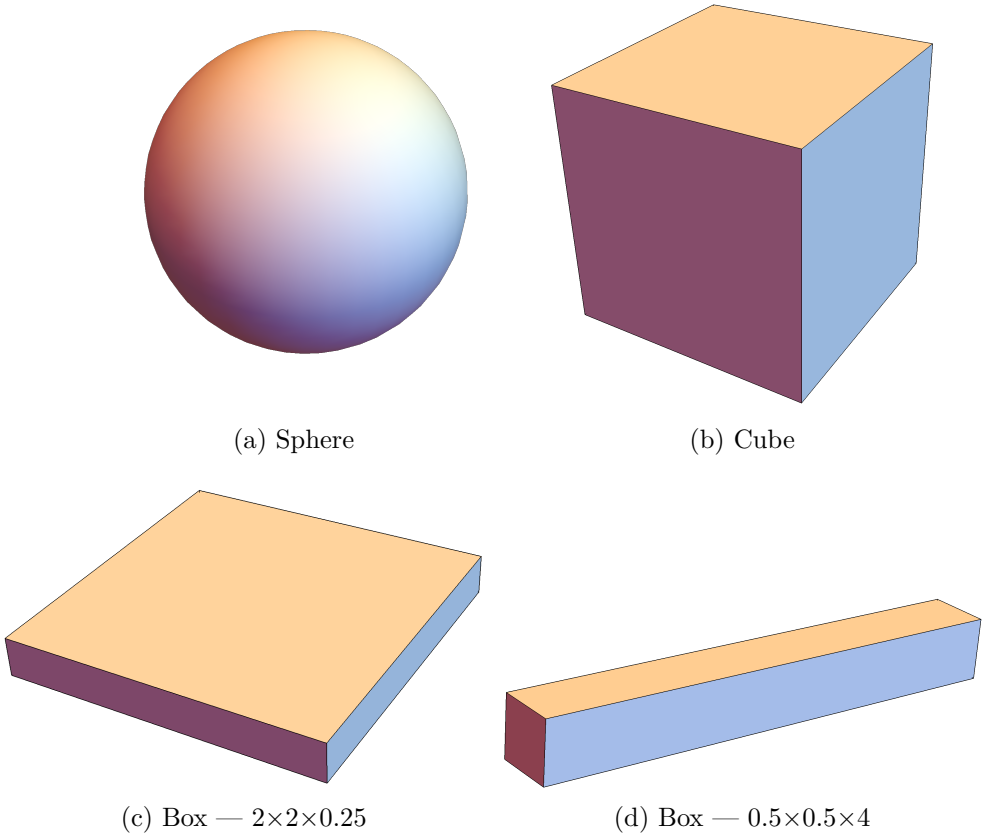
Each probability distribution function requires a normalization factor which is treated as a separate Monte Carlo integration. We randomly sample from all possible orientations uniformly given a position. The normalization factor is calculated until it has converged within an error of less than 0.01%. In practice, it converges quickly.

Due to the high variance in gains at high antenna counts, we use 200K samples to calculate the expectation. Positions are sampled within the space with uniform probability. The probability density function is sampled using rejection sampling. We sample 600 points and average across all of them. All numbers are represented using double precision floating point.

For the choice of antenna arrays, we only consider planar antenna arrays and simulate every square array up to a 40×40 array using $\lambda/2$ spacing. The rectangular arrays are chosen based on those that can evenly divide the square array.

4.6 Results

Four different finite extents are explored, Figure 4.6: Sphere, Cube, Box 2×2×0.25, and Box 0.5×0.5×4. The results for the sphere and rectangular cuboids are presented separately. The legends use *Rect n* or *Rect n*×. These refer to a rectangular array whose one axis is fixed to being of n antenna elements. Note that when referring to rectangular arrays in this section it *excludes* the linear array.



(a) Sphere

(b) Cube

(c) Box — 2×2×0.25

(d) Box — 0.5×0.5×4

Figure 4.6: Different Finite Extents

Sphere

The results for the sphere are shown in Figure 4.7 comparing the linear array against other rectangular arrays. As expected the linear array is strictly better for all possible antenna counts. Interestingly, the square array's performance has an oscillatory pattern.

To further investigate this oscillation, let us look at the derivative of the interference with respect to antenna count, Figure 4.8. Note that the y-axis of this graph does not have a well defined unit as only things relevant to changing antenna count were simulated, so

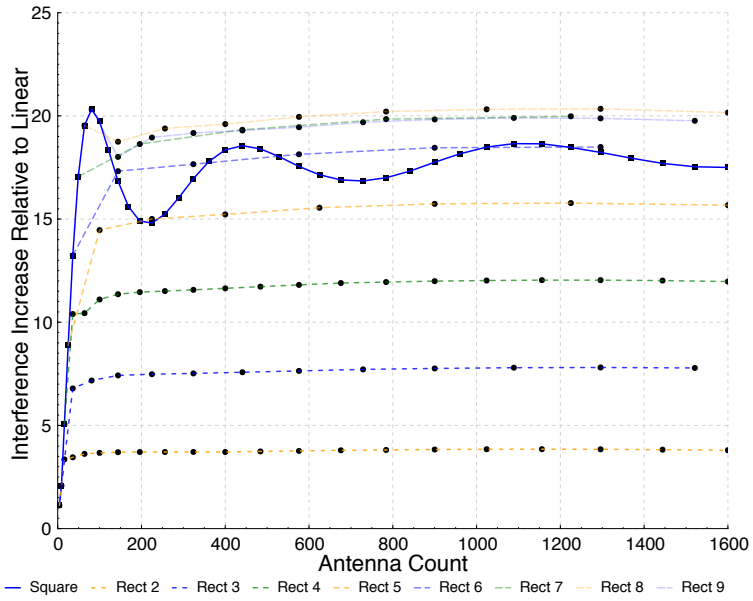


Figure 4.7: Interference increase by using rectangular arrays compared to a linear array inside a sphere.

things such as path loss are not shown. In this graph, it is clear that one can attribute the oscillation to the square array and not the linear array. In addition, it appears to be unique among the rectangular arrays.

Furthermore, it would be nice to know what is causing this behavior. To gain a better idea, the performance of the arrays was examined using only the 3db-width mainlobes, Figure 4.9. Here the oscillation has disappeared. Therefore, the cause is directly due to the sidelobes. Most likely due to the fact that sidelobes scale unusually if you constrain the array to be a square.

Another interesting thing to note comes from comparing the two graphs: mainlobe and sidelobes, Figure 4.7; and mainlobe only, Figure 4.9. Figure 4.10 shows the percentage of interference that comes from mainlobe to mainlobe. For very low number of antenna counts, the majority of the interference increase can be attributed to the mainlobe. However, after about 100 antenna elements, sidelobes start to bear more responsibility for the interference. Another interesting property can be seen that for large antenna counts linear arrays have about half of their interference come from mainlobe to mainlobe, but for other arrays, less interference is coming from the mainlobe to mainlobe and more from other configurations.

The percentage of interference coming from mainlobe to sidelobe and sidelobe to sidelobe are shown in Figure 4.11 and Figure 4.12, respectively. Interestingly, the contribution of the interaction between mainlobes and sidelobes converges for each antenna array geometry (see Appendix C for each array's convergence). Figure 4.14 shows what the expected interference contributions converge to for each array. 1296 antenna elements was chosen for all rectangular arrays expect arrays 5x and 7x that uses 1225 antenna elements (due to divisibility).

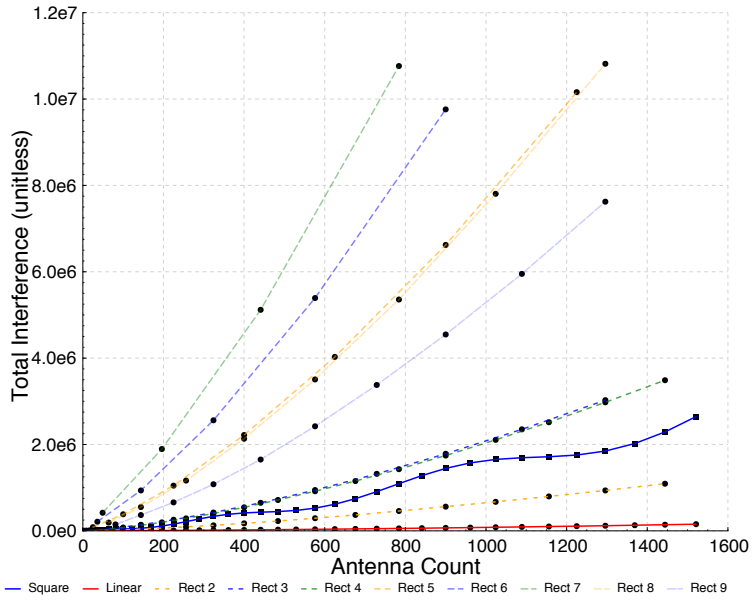


Figure 4.8: Derivative of the interference with respect to antenna count inside a sphere

For all antenna arrays, it seems that about 40–45% of the expected interference is from mainlobe to sidelobe. However, the antenna array geometry plays a larger role in mainlobe to mainlobe and sidelobe to sidelobe. For antenna arrays that are close to being linear arrays, the majority of the interference is from mainlobe to mainlobe and very little interference comes from sidelobe to sidelobe. The opposite is true for arrays that have more than have 5 elements in both dimensions. It is not clear as to why antenna array geometry plays such a large role in this case and should be investigated in future work.

Rectangular Cuboid

Figure 4.15 shows the results for a cube. Here we see that the linear array is universally better. Compared to using a square array, we can get close to a 40x improvement in interference reduction.

As discussed in Section 4.4, we should expect that the linear array will outperform other rectangular arrays independent of the ratio of the space. Fig. 4.16 shows the interference increase of using various rectangular arrays compared to a linear array for a rectangular cuboid of size $0.5\times0.5\times4$. Likewise, in Fig. 4.17, you can see the interference increase for size $2\times2\times0.25$.

As can be seen in the figures, the linear array is always strictly better. In addition, when the space under consideration is not a cube, using a square antenna array becomes an even worse choice of array geometry. As discussed earlier, this is due to the bias of wanting to point along the longest axis compared to the shortest axes.

Lastly, we examine the performance of individual node positions. We take as a case

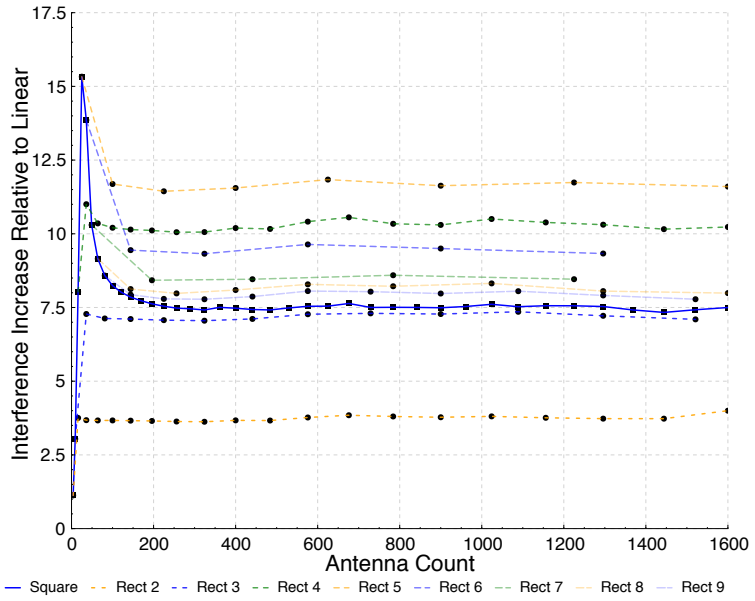


Figure 4.9: Interference increase by using rectangular arrays compared to a linear array inside a sphere. 3dB-width mainlobes only.

example a comparison between the 16×16 array and 1×128 array, Fig. 4.18. The large spread of interference increases in Fig. 4.18 can be attributed to undersampling in the Monte Carlo integration. We observe that for high antenna counts there does not appear to be a significant dependence of position on performance. Therefore, only the nodes as a function of the x-axis are depicted, Fig. 4.18a. For the 2×2×0.25, we examine the shortest axis (0.25) as the other two axes are symmetric. Once again, there does not appear to be a significant difference in performance for the linear vs square in this axis as well, Fig 4.18c. It is only for the 0.5×0.5×4 box that a performance difference based on position is seen, Fig. 4.18b. Looking at the longest axis (4), we see that the square array experiences more interference in the middle than toward either end.

Overall, the capacity improvement is proportional to the original SINR of the system. If we assume that interference is significantly higher than the noise, then we can easily derive the system performance improvement. Let I_0 be the linear array's interference; S be the signal; and α be the interference increase.

$$Improvement = 1 + \frac{\log_2 \alpha}{\log_2 \frac{S}{I_0}} \tag{4.19}$$

From these experiments, I believe that there exists a proof that shows linear arrays are optimal among all planar antenna array geometries of the same aperture, given our assumptions. However, it is left for future work.

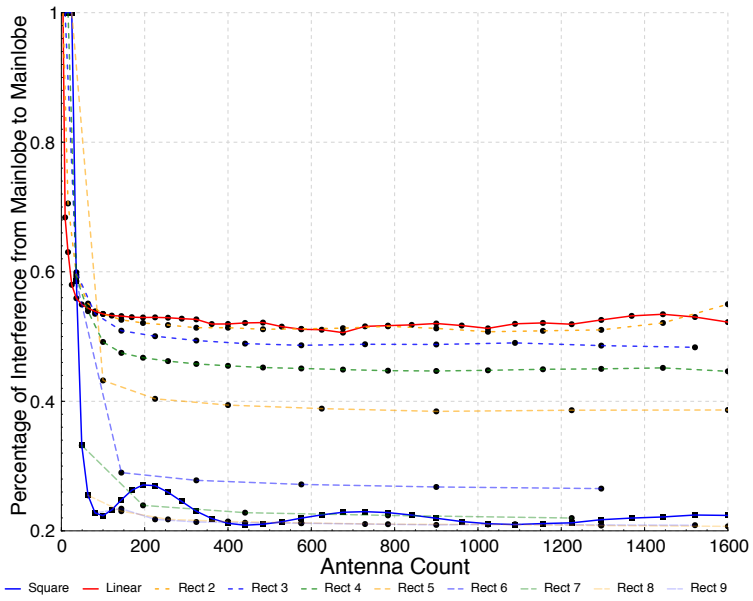


Figure 4.10: Percentage of Interference due to Mainlobe to Mainlobe.

4.7 Summary

I have shown that systems with linear arrays have less interference when boundary effects are taken into account independent of the number of antenna elements. In particular, I have shown through analytical modeling and numerical integration that linear arrays can be a significant improvement over other planar antenna array geometries, almost 40x when considering a cube and significantly more for other operating spaces.

In addition, expected sources of interference were presented. For all rectangular arrays, 40% of expected interference comes from mainlobe to sidelobe (and vice versa). Mainlobe to mainlobe interference is most likely to occur for arrays that are similar to linear. However, when the arrays become more square-ish, it is more likely for the interference to occur sidelobe to sidelobe (and vice versa) compared to mainlobe to mainlobe.

There remain many possible extensions to this work. Different antenna array geometries (other than rectangular) should be considered. Though, I anticipate rectangular arrays to be optimal due to the reasons discussed in Section 1.3. Antenna spacing should also have an effect. Due to preliminary work during the interference mitigation research, I believe that λ spacing may actually be optimal compared to $\lambda/2$. Lastly, there is likely an analytical proof showing that linear arrays are strictly optimal under the assumptions that were outlined.

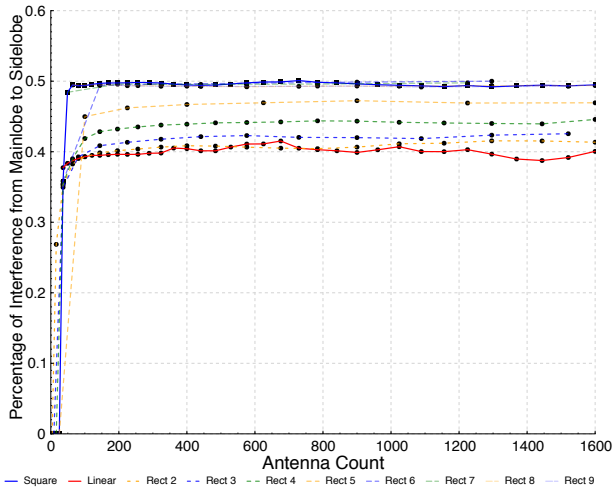


Figure 4.11: Mainlobe to Sidelobe (and vice versa).

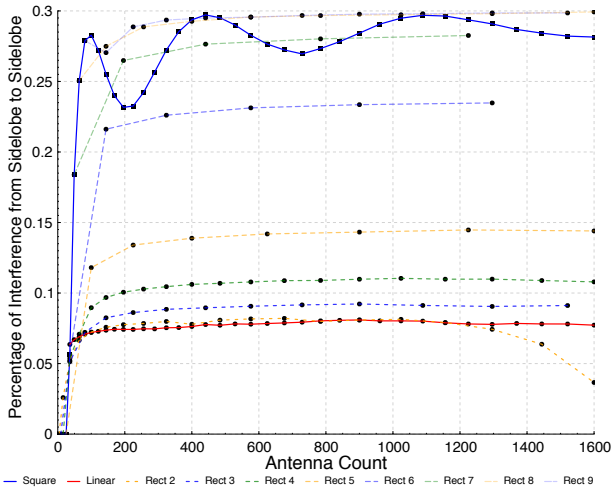


Figure 4.12: Sidelobe to Sidelobe.

Figure 4.13: Percentage of Interference in Sphere.

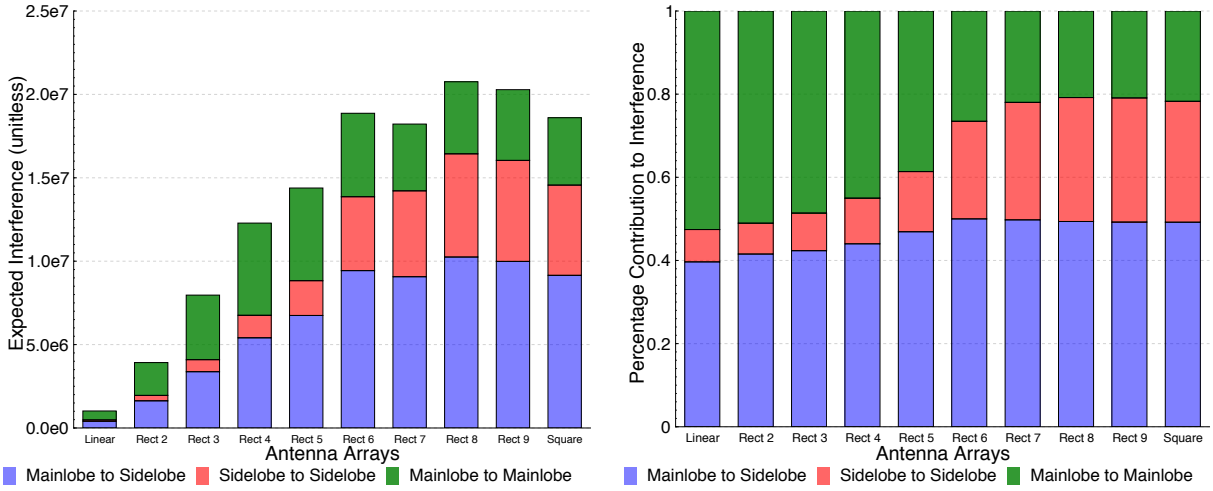


Figure 4.14: Converged Percentage of Expected Interference in Sphere.

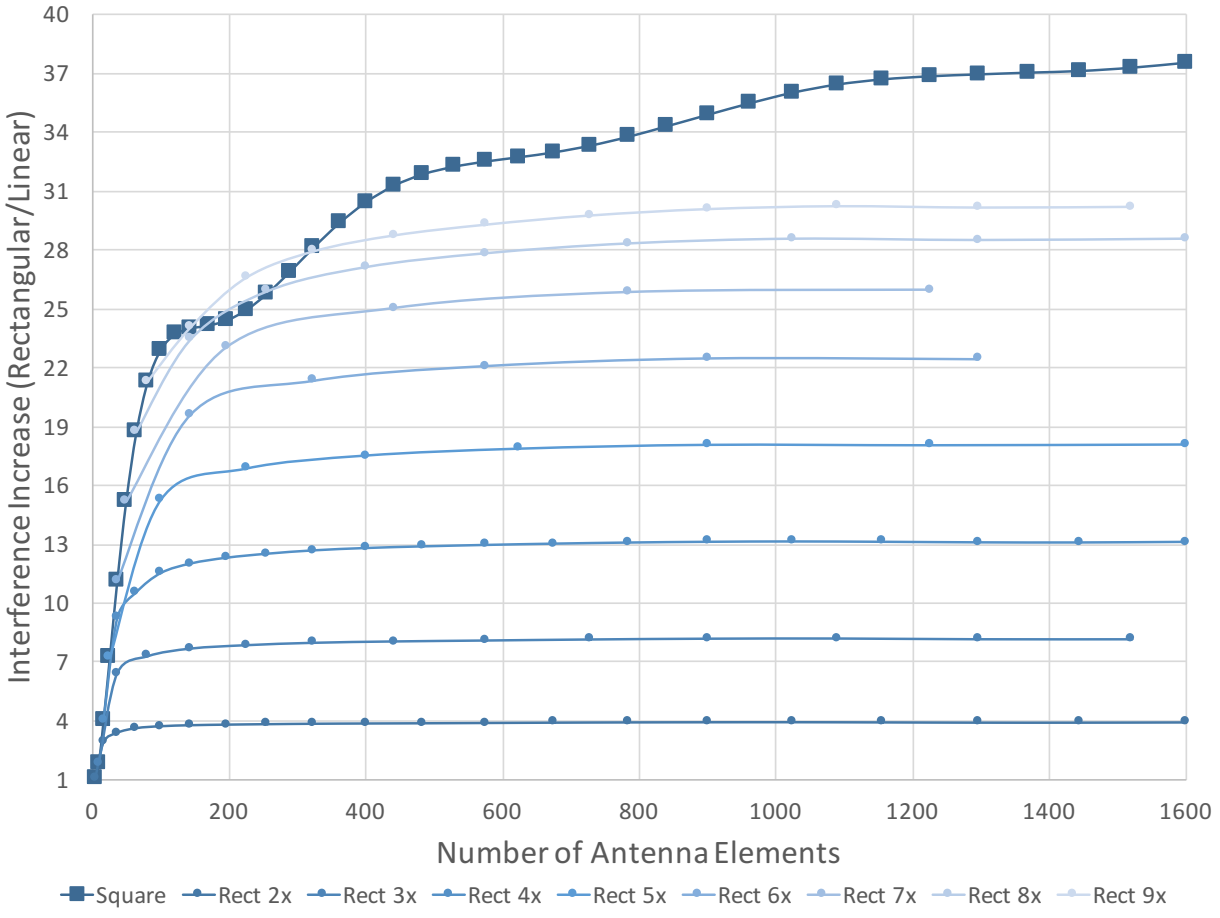


Figure 4.15: Interference increase by using rectangular arrays compared to a linear array inside a cube.

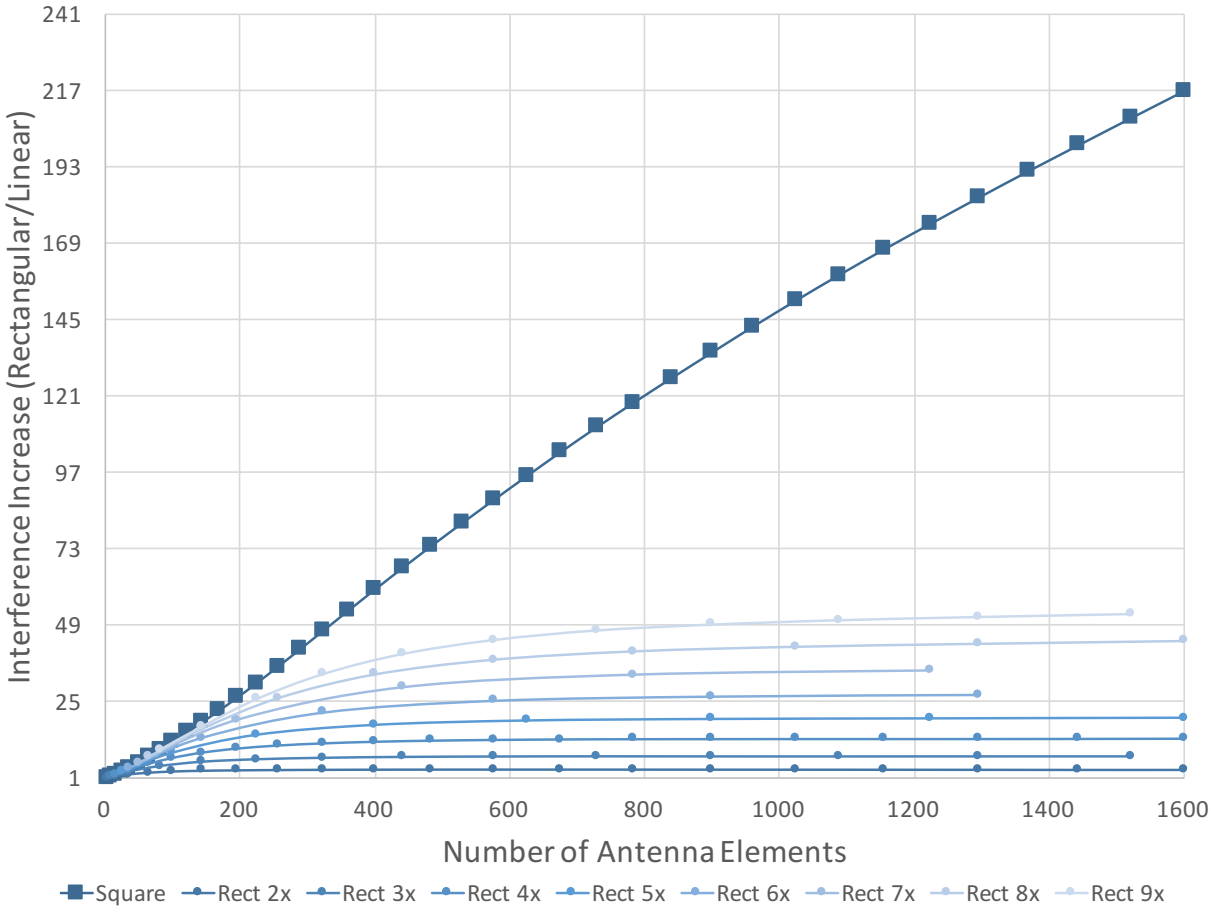


Figure 4.16: Interference increase by using rectangular arrays compared to a linear array inside a $0.5\times0.5\times4$ box.

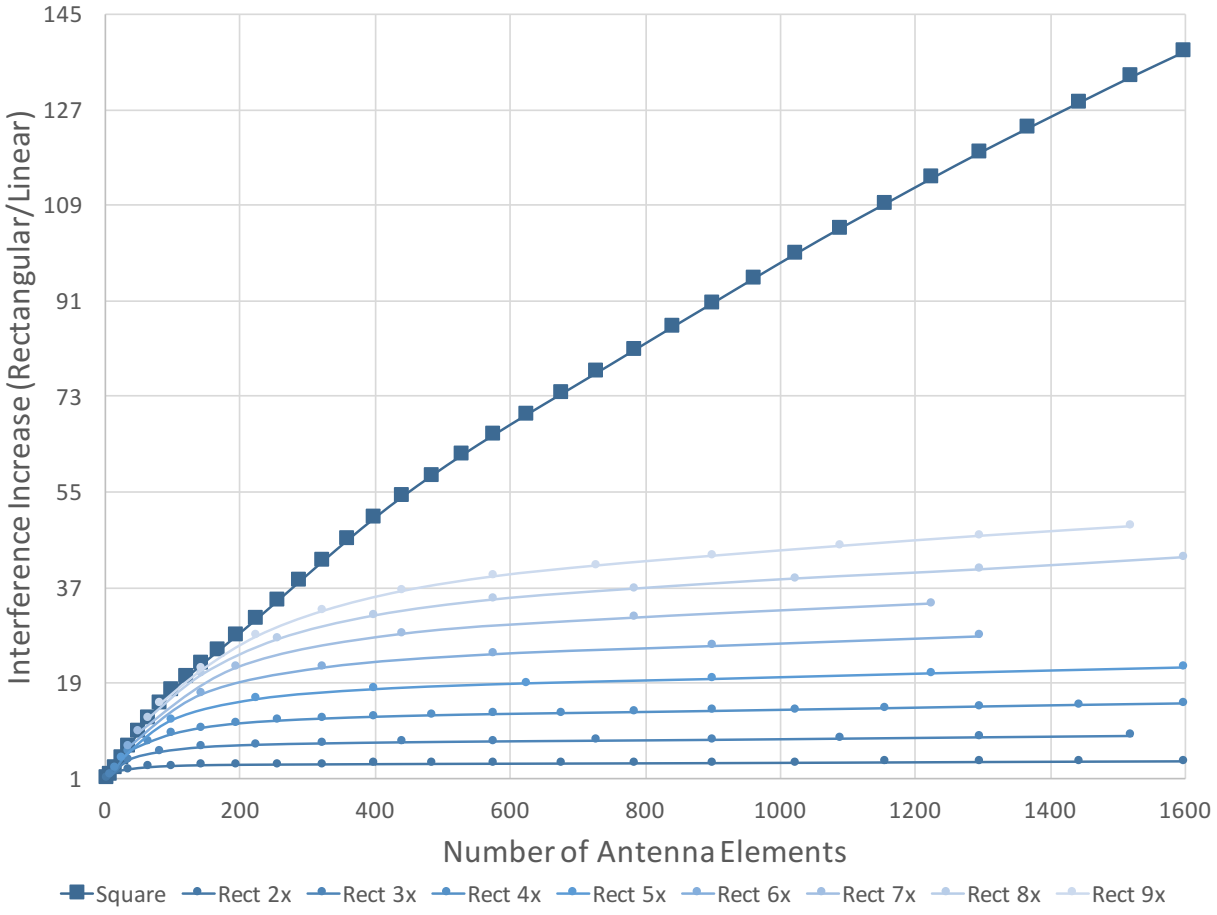
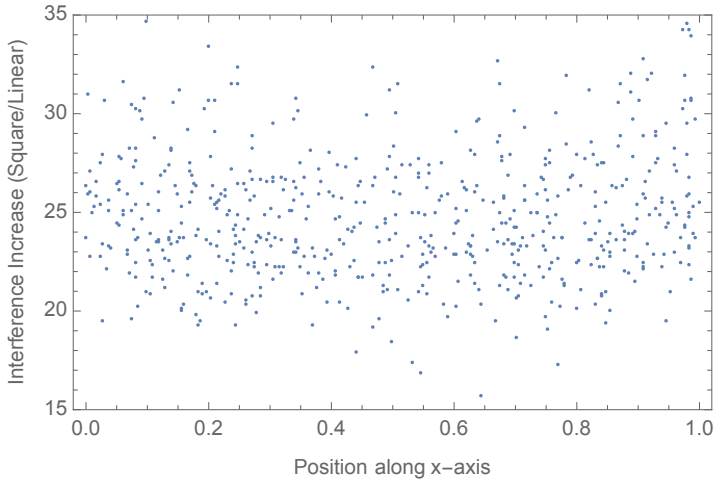
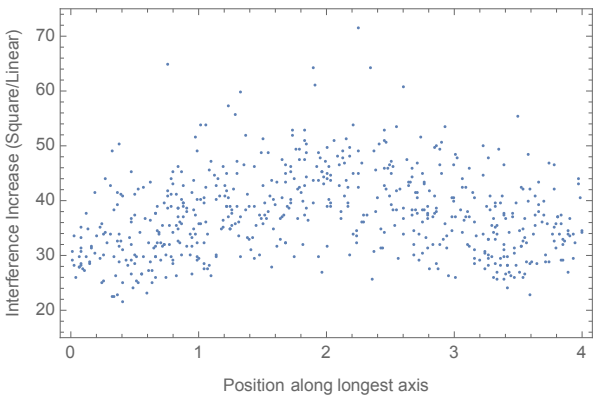


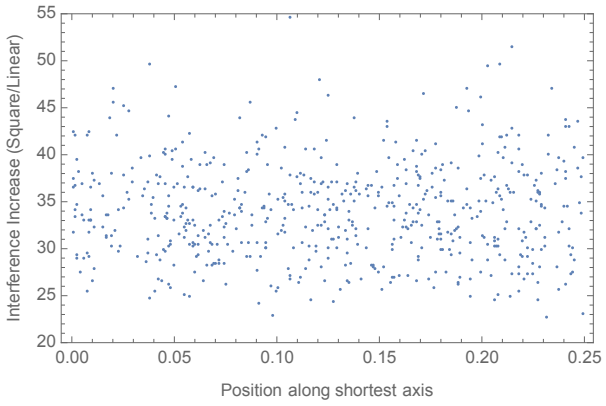
Figure 4.17: Interference increase by using rectangular arrays compared to a linear array inside a $2\times2\times0.25$ box.



(a) Cube



(b) 0.5×0.5×4 Box



(c) 2×2×0.25 Box

Figure 4.18: Interference increase comparing a 16×16 array and 1×256 array. Each point is a unique node position.

Chapter 5

Listen-only Scheduling

As mentioned earlier in the introduction, a major challenge in the control of wireless networks with steerable directional radios is coordinating the transmission and receiving directions of nodes when there is a possibility for one node to talk to more than one node in the network using the same antenna array. In addition, there is an issue of deafness such that one node may be unaware of when other nodes are transmitting as well as the exposed node problem where a node may falsely believe that it cannot transmit. Because of this, a new link management scheme is proposed where each node determines its listening schedule independently called Listen-only Scheduling. Transmission is not explicitly scheduled. This has two major benefits. Scheduling only the receivers leads to control overhead proportional to the number of nodes instead of the square of the number of nodes as is the case with joint transmission. Second, scheduling itself removes the issue of deafness and exposed node.

This chapter examines the performance of Listen-only scheduling in terms of latency as well as throughput. Two adaptation algorithms are proposed to optimize these metrics, traffic-based and queue-based. These algorithms are evaluated against other link management algorithms. In addition, there is an evaluation of the maximum throughput that can be achieved for listen-only scheduling and how close the algorithms can reach this upper bound.

The system architecture matches the system model (Chapter 2). However, the channel model is different and described in the evaluation, Section 5.5.

5.1 Previous Work

There are currently two widely used MAC industry protocols for mmWave, both for 60GHz. The first is WirelessHD, which aims to provide connectivity between home entertainment devices such as televisions and gaming devices [70]. The second is WiGig, which is the industrial initiative of 802.11ad [67]. 802.11ad builds from the current MAC protocols developed in previous 802.11 standards and adapts them for use in 60GHz and directional beams [30].

The majority of previous work including industry protocols relies on omni-while-idle

receiving [70, 30, 6, 7]. The idea being that when a node is not communicating it should listen in all directions for transmissions. Omni-while-idle is subject to collisions, and we study this effect by using Directional Slotted ALOHA. In order to avoid collisions, a pseudo-omni mode where the receive beam is rotated around in a circle has been proposed in some MAC protocols [42, 60, 34]. However, the receive beam must be wide enough in order to not incur too high of an overhead of scanning all directions. Unfortunately, wide beams are subject to interference in dense environments, and the benefits over omnidirectional are less significant. If narrow beams are used, a significant overhead will be incurred in simply switching the beam among all of the possible directions.

As far as coordination, the majority of previous work and industry protocols rely on centralized control [70, 30, 6, 7]. There existed an industrial protocol that was distributed called ECMA-387 [18]. However, to date, it has not seen any widespread adoption compared to WirelessHD or WiGig. Singh et. al. proposed a distributed slotted control approach where nodes remember in which slots there are collisions and no collisions [55]. If a transmission is successful, that slot is then reserved for that node. Otherwise, the slot is blacklisted for a time. The approach allows for adaptation in the schedule based on the demands of the neighbors. However, it relies on omnidirectional receive to establish a schedule and is random in nature. My approach is deterministic and does not rely on omnidirectional receive to create a schedule. Determinism offers the advantage of making coordination among nodes easier.

5.2 Mechanism

A listen-only schedule is a finite time-slotted schedule that indicates when a node will listen to which neighbor. Each node maintains its own *independent* schedule of when and to whom it will be listening. For every node n, its neighbors, $neigh(n)$, are given a list of time slots, $[TimeSlot]$, over some interval, I. Each $TimeSlot$ can be one of three states, Table 5.1. How $[TimeSlot]$ is encoded is not specified and should be based on what is assumed about synchronization.

Time Synchronization

At the minimum each node needs to be time synchronized with each of its neighbors such that the node can determine the relative time it needs to transmit. For example, consider a network of three nodes all able to transmit to each other each with their own notion of time, Figure 5.1. The simplest case is that every node is time synchronized to every other node (i.e. $T_A = T_B = T_C$). However, this is a very strong requirement that does not scale well as the number of nodes increases due to the overhead of synchronization.

However, because each node maintains its own time schedule, the requirement for global synchronization is sufficient but not necessary. A simpler requirement is maintaining separate synchronization for each neighbor called local synchronization. In this way, each node

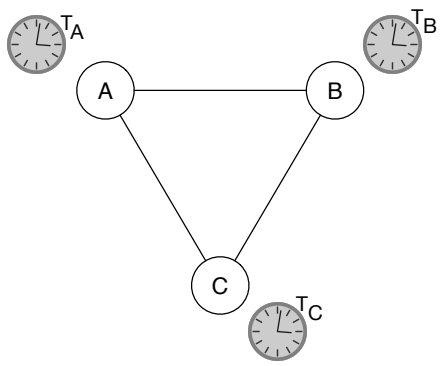


Figure 5.1: Three node topology with time

maintains a function of how a neighbor's time relates to its own time. Using the example of three nodes, node A would have two functions for each of its neighbors, B and C. Function $f : T_B \rightarrow T_A$ takes in a time from the time domain of node B and returns the relative time in the time domain of node A. The advantage of this synchronization strategy is that the synchronization requirement for communication scales with the number of neighbors and not with the size of the network. The disadvantage is that each node requires storage proportional to the number of neighbors. Though, the storage requirement is likely to be small.

For the purposes of evaluation, though, only global synchronization is examined.

Slot Types

The slots can be one of three types, 'One,' 'None,' or 'Any,' relating to whom the node is listening.

One(x)	Listening exclusively to x
Any	Listening for any neighbor
None	Not listening to anyone

Table 5.1: Time slot types

The 'One' type indicates that the node is listening exclusively to one neighbor. This type contains an additional field corresponding to the node identifier (and also possibly the direction necessary for reception). When the node is in a 'One' type, its receive beam is pointed at the corresponding neighbor. The receiver being in this state creates a transmission opportunity for the neighbor intended to be used for regular data transmissions, schedule updates, and beamforming adaptation and refinement between two nodes. The decision to make a transmission is made autonomously from the transmission opportunities of the relevant neighbor.

The 'None' type indicates that the node is listening to no one. When the node is in the 'None' state, the node is unable to listen to any nodes that are in the network. The interface might be switched off for the sake of energy saving.

Lastly, the 'Any' type indicates that the node is either listening omnidirectionally or simulating omnidirectionality (see Section 5.2). In this way, the node is listening to any neighbor that attempts to transmit to it. The 'Any' type is intended to be used for the mechanisms necessary for discovery. Though, it could also be used for carrier sense link management as well.

Each node determines its listen-only schedule independently of the scheduling decisions of its neighbors. It does not forbid cooperation but instead makes cooperation optional. A neighboring node x only needs to know at what times a node is in state One(x). Nodes share with its neighbors its listen-only schedule corresponding to a block of time in the future and a time at which the neighbors should start using the schedule. Listen-only schedules are shared with the neighbors using the neighbor's transmission opportunities (i.e. the 'One' type). Note that the amount of information necessary for a listen-only schedule update is directly related to how frequently updates are performed. This amount of information will dictate how the update is implemented, either as part of normal data packets, a separate packet, or a reserved time as part of a slot.

Simulating Omnidirectionality

If one does not have a priori knowledge of the location of new nodes, it is necessary to be able to both transmit and receive in all directions. If nodes have planar antenna arrays as are common in mmWave (see Section 1.2 for a discussion on this), then it may be necessary to have multiple antenna arrays to have full coverage of the space of interest if there are not sufficiently good reflectors. In addition, the antenna arrays themselves are directional and therefore are deaf to some directions. To compensate for this, one must simulate omnidirectionality.

To simulate omnidirectionality for an antenna array, one uses multiple beam configurations. The specific technique may vary depending on the transmitter or the receiver. For the transmitter, a simple technique is to point the beam at different directions over a period of a fixed amount of time such that all directions are covered. This technique is commonly called beam sweeping and is implemented in 802.11ad [30]. Note that when simulating omndirectional transmission the packet must be repeated for every direction.

For the receiver, one may use the same strategy as the transmitter. However, there exists an alternative strategy that scales linearly with the number of antennas. One such strategy would be if you have n antennas to use a $n \times n$ complex Hadamard matrix [58, 28], Section 1.3. Applying the inverse of the matrix to the response allows one to determine the direction of the signal assuming it is the strongest one. If it is not the strongest, coding is also necessary. Note that when simulating omnidirectional receiving, a sufficiently long preamble is necessary to determine the direction.

In either scenario, if the simulation of omnidirectionality is judged to take too long, less antennas may be used to cover larger angles at once. This comes at the expense of less directionality and gain, overall decreasing the amount of coverage and potentially increasing the amount of interference.

Discovery

The process of adding a new node to an existing network, called discovery, consists of three steps: beacon, beacon response, and schedule update. A new node hears a beacon from an existing node. The new node responds with a beacon response to the existing node. The existing node sends to the new node its listening schedule. For convenience, when necessary, a new node will be called Nancy, and an existing node will be called Eve.

The first step is beaconing. Each node in an existing network reserves some small amount of transmission time for beaconing by beam sweeping. The precise amount of time may vary depending on the requirements of the network (e.g. how quickly must new nodes be added to the network?) and the density of the network (e.g beacon less if the node already has many neighbors). Each node that beacons also reserves some number of 'Any' slots that can be used to hear a beacon response from a new node. A beacon consists of a node identifier, the direction used for the beacon, information necessary for time synchronization, and at least one time at which the new node may respond with a beacon response. New nodes that are not part of the network have a listen-only schedule of all slots being 'Any'. New nodes employ the omnidirectional receiving strategy described in Section 5.2. This strategy also gives the direction for the new node to the existing node. Existing nodes in an existing network may also reserve 'Any' time slots to discover new nodes that may have been missed.

Following a beacon reception, Nancy needs to transmit a beacon response during one of Eve's 'Any' slots that it reserved for this purpose. When Eve receives a beacon response from a previously unknown node (such as Nancy), Eve schedules Nancy as part of its listen-only schedule. The beacon response contains the new listen-only schedule from the previously unknown node, Nancy, and information necessary for time synchronization. Eve that sent the original beacon now knows how to communicate with Nancy; both the direction (due to the technique described in Section 5.2) and the timing is known.

Following the beacon response, Nancy must be given Eve's listening schedule that incorporates Nancy in the schedule. This is done on a regular transmission 'One' time slot from Nancy's listen-only schedule. After this is done, both nodes exist in each other's schedules and directional transmission and receiving should be used.

5.3 Listen-only Schedule Conflict

Note that because nodes determine their listening schedules independently, there is a new possible source of performance loss that does not exist when both sending and listening are

scheduled together. A listening schedule conflict happens when two nodes listen to the same node at the same time, and the node has packets for both of them.

To analyze the likelihood of a listen-only schedule conflict, assume each node communicates with each neighbor equally, and that schedules are randomly ordered with only exclusive listening as part of their schedule.

Probability of the Same Slot

First we will calculate what is the likelihood of two nodes giving a slot to the same node. Let $nbrs(x)$ be the set of neighbors of node x, and $slots(x)$ be the slots of the schedule of node x. From our assumptions, we can conclude that

$$\forall s \in slots(x), i \in nbrs(x). \Pr[s = i] = \frac{1}{|nbrs(x)|} \tag{5.1}$$

For a given neighbor pair (i, j) for a given node, the probability that i and j will pick the same slot, $SameSlot(i, j)$ is given as

$$\Pr[SameSlot(i, j)] = \frac{1}{|nbrs(i)|}\frac{1}{|nbrs(j)|} \tag{5.2}$$

Let $allPairs(x)$ be the set of all combinations of neighbor pairs of node n.

$$allPairs(x) = \{\forall i \in nbrs(x), j \in nbrs(x), i \neq j.\{i, j\}\} \tag{5.3}$$

The probability of the same slot given to n, $AnySameSlot(x)$, is then:

$$\begin{aligned} &\Pr[AnySameSlot(x)] \\ &= \sum_{\{i,j\} \in allPairs(x)} \Pr[SameSlot(i, j)] \\ &= \sum_{\{i,j\} \in allPairs(x)} \frac{1}{|nbrs(i)|}\frac{1}{|nbrs(j)|} \end{aligned} \tag{5.4}$$

Obviously, if the number of neighbors does not increase linearly for all nodes near a given node, then as more nodes are added, the probability approaches 1. However, this is very unlikely as this would imply a long chain of nodes that are very far apart from each other and would require infinite space in the limit. Therefore, it is reasonable to assume that as one node increases its neighbors, the other neighbors also see a linear increase in the number of their neighbors. It is also then reasonable to analyze the situation where everyone has the same number of neighbors, δ, and see the effect of the change of probability as a function of the number of neighbors. This gives an intuition of what happens as the density of nodes increase.

$$\Pr[AnySameSlot(x)] = \binom{\delta}{2}\frac{1}{\delta^2} = \frac{\delta - 1}{2\delta} \tag{5.5}$$

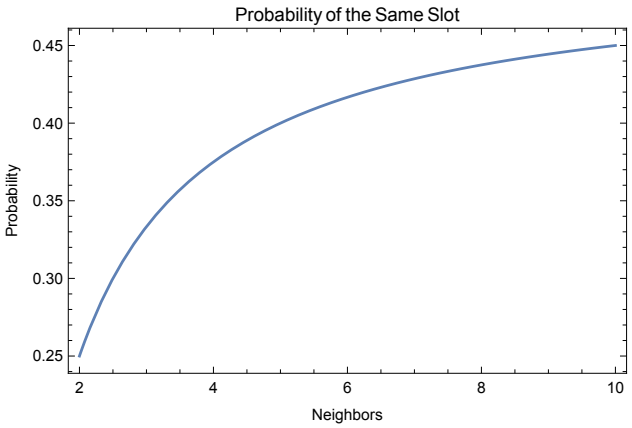


Figure 5.2: Probability of Same Slot

This is shown graphically in Figure 5.2.

If we take the limit as δ goes to ∞, we see that the probability is 0.5.

$$\lim_{\delta \to \infty} \frac{\delta - 1}{2\delta} = \frac{1}{2} \tag{5.6}$$

Therefore a reasonable bound is to say that the probability of the same slot is expected to be no greater than 0.5.

Probability of a Conflict

To analyze the probability of a conflict, assume further that packet next hops are i.i.d.; there is an output queue per neighbor; and the set of queues is not strict FIFO. Let $nextHop(p)$ be the next hop of packet p. Let $buf(x)$ be the set of packets buffered in node x. From packets next hops being i.i.d.:

$$\begin{aligned} &\forall p \in buf(x), i \in nbrs(x). \\ &\quad \Pr[nextHop(p) = i] = \frac{1}{|nbrs(x)|} \end{aligned} \tag{5.7}$$

If we have $|buf(x)|$ packets, then the probability that there does not exist a packet in the buffer for a neighbor i is given by:

$$\begin{aligned} &\forall p \in buf(x), i \in neighbors(x) \\ &\quad \Pr[\nexists p \in buf(x).nextHop(p) = i] \\ &\qquad = (1 - \Pr[nextHop(p) = i])^{|buf(x)|} \\ &\qquad = \left(\frac{|nbrs(x)| - 1}{|nbrs(x)|}\right)^{|buf(x)|} \end{aligned} \tag{5.8}$$

Let $NextHop(i,j)$ be the event that are two packets buffered that are $nextHop(i)$ and $nextHop(j)$, respectively.

$$\begin{aligned} NextHop(i,j) = \\ &(\exists p \in buf(x).nextHop(p) = i) \wedge \\ &(\exists p' \in buf(x).nextHop(p') = j) \end{aligned} \tag{5.9}$$

The probability of this event is then:

$$\begin{aligned} &\forall i \in nbrs(x), j \in nbrs(x), i \neq j \\ &\quad \Pr[NextHop(i,j)] \\ &\quad\quad = \Pr[\exists p \in buf(x).nextHop(p) = i] \cdot \\ &\quad\quad\quad \Pr[\exists p \in buf(x).nextHop(p) = j] \\ &\quad\quad = (1 - \Pr[\nexists p \in buf(x) = i]) \cdot \\ &\quad\quad\quad (1 - \Pr[\nexists p \in buf(x) = j]) \\ &\quad\quad = \left(1 - \left(\frac{|nbrs(x)| - 1}{|nbrs(x)|}\right)^{|buf(x)|}\right)^2 \end{aligned} \tag{5.10}$$

Let $Conflict(x)$ be the event of a listening schedule conflict.

$$\begin{aligned} \Pr[Conflict(x)] = \\ \sum_{\{i,j\} \in allPairs(x)} \Pr[SameSlot(i,j)] \Pr[NextHop(i,j)] \end{aligned} \tag{5.11}$$

This reduces to the following:

$$\begin{aligned} \Pr[Conflict(x)] = &\left(1 - \left(\frac{|nbrs(x)| - 1}{|nbrs(x)|}\right)^{|buf(x)|}\right)^2 \cdot \\ &\sum_{\{i,j\} \in allPairs(x)} \frac{1}{|nbrs(i)||nbrs(j)|} \end{aligned} \tag{5.12}$$

Again, we will let δ be the number of neighbors per node. We will also let b be the number of packets buffered in the node. From this, we get

$$\Pr[Conflict(x)] = \binom{\delta}{2} \frac{1}{\delta^2} \left(1 - \left(\frac{\delta - 1}{\delta}\right)^b\right)^2 \tag{5.13}$$

Let $b = c$, where c is a constant that does not scale with the number of neighbors. This case is shown in Figure 5.3 for a number of constants. As can be seen as the number of neighbors increases, the probability approaches zero.

Now, let $b = m\delta$, where m represents some weighting of the number of packets compared to the number of the neighbors. This case is shown for various m's in Figure 5.4. Figure 5.5

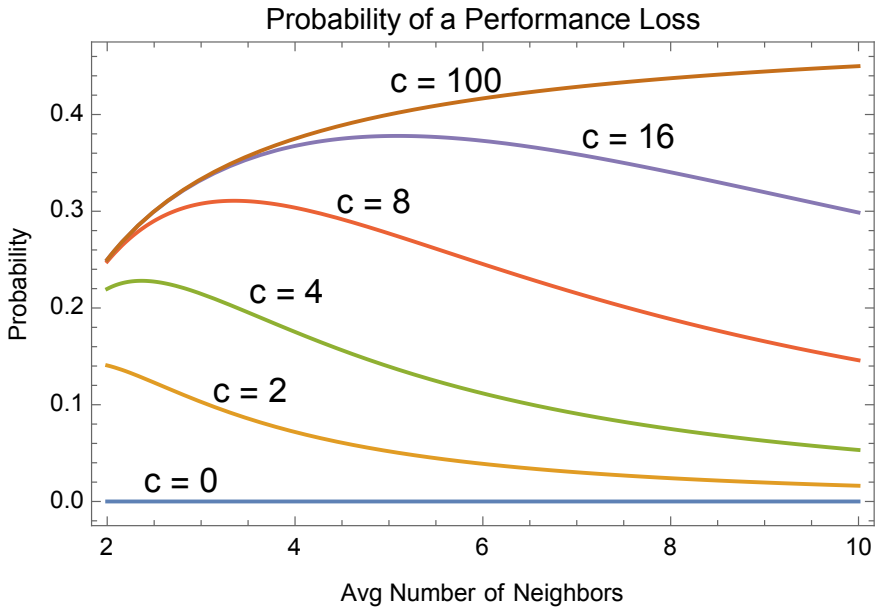


Figure 5.3: Probability of Conflict: Constant Buffering

shows the case where δ is kept fixed and m is varied. Taking the case of $m = 1$, we see that as the number of neighbors is increased the probability approaches approximately 0.1998. From our analysis, assuming packet next hops are i.i.d. random schedule ordering with no head of queue blocking, we have concluded that if there are a small number of packets in a given node relative to the number of neighbors, there is a low probability of a listen-only schedule conflict. In practice, nodes usually have a low number of packets in their queues except when the network is being used at high utilization. In addition, in real networks, packets' next hops are not i.i.d. and instead show a strong bias to one neighbor or another due to differences in capacity of links or the fact that one neighbor is closer to a gateway than another neighbor. This fact means that for dynamically adapted schedules, certain neighbor nodes will be favored over others and, the number of neighbors of interest will be significantly less than all possible neighbor, and hence the likelihood of a collision will be less.

5.4 Schedule Adaptation

Adaptation is an iterative process that uses neighbor's information to improve the efficiency of the schedule. An efficient schedule is one where neighbors that have data to send are able to send it in the minimum time achievable. Each listen-only schedule can be changed independently of other schedules, and this allows for completely distributed adaptation of schedules to changes in the network. Schedule changes can be made locally using only the information that is received as part of the traffic. Two possible sources of information for

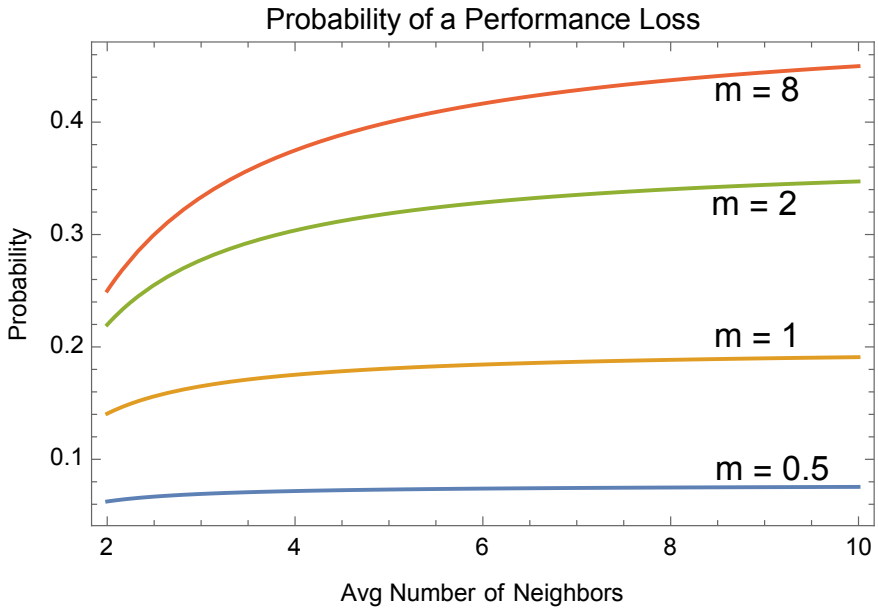


Figure 5.4: Probability of Conflict: $b = m\delta$. δ varied.

adaptation are considered: the number of packets received over time and the state of the neighboring queues.

Problem Definition

At a receiver, the neighborhood consists of N transmitters. Each transmitter $i \in [1, N]$ has a demand of the receiver, s_i slots. A slot is a fixed amount of time. The receiver, as part of its schedule, gives a transmitter i an allocation of t_i slots. The receiver observes the usage of the channel, τ_i, as a function of s_i and t_i.

$$\tau_i = \begin{cases} s_i & t_i \geq s_i \\ t_i & t_i < s_i \end{cases} \tag{5.14}$$

Let the total number of slots for a receiver's schedule be M. M is chosen to be sufficiently long to keep schedule updates from being too frequent and to accommodate enough granularity for the desired allocations. All s_i and t_i are constrained to be greater than 0 and less than M. Furthermore, note that $\sum t_i \leq M$. Notice that there exists a schedule that can satisfy all demands if and only if $\sum s_i \leq M$. If $\sum s_i \geq M$, then not all demands can be satisfied, and one must choose a fairness algorithm to decide the share of demands for individual neighbors that will be satisfied.

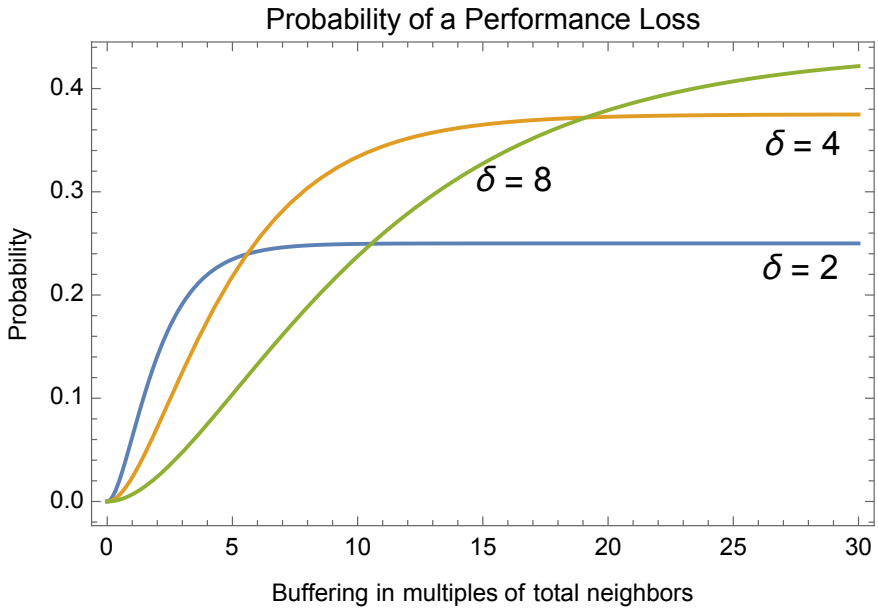


Figure 5.5: Probability of Conflict: $b = m\delta$. m varied.

Traffic-based Adaptation

For the traffic-based adaptation, the receiver only uses t and τ. The receiver does not have access to s. A natural algorithm to use is the additive-increase, multiplicative decrease with a multiplicative-increase when utilization is very low. The adaptation algorithm is described below.

```
while (True):
    for i in neighbor_transmitters:
        if τ_i == t_i:
            if sum(t_i) << M:
                r_i = min(α*t_i,1)
            else:
                r_i = t_i + γ
        else:
            r_i = max(β*t_i, τ_i, min_alloc)
    t = min_max_fairness(r,M)
    sched = make_schedule(t, update_freq)
    synchronize_schedule_with_neighbors(sched)
    wait(update_freq)
```

This algorithm states that for every transmitter among the node's neighbor set, determine the number of slots used by the transmitter. If the transmitter is using all of the allocated slots, increase the allocation by multiplicative weight α (always $\alpha > 1$) if the number of

slots is below the additive-multiplicative threshold, (sum(t_i) << M). Otherwise, increase the number of slots by an additive of γ (always $\gamma > 0$). If the neighboring transmitter is not using all of its allocation, then the allocation that it receives is decreased by multiplicative weight β (always $0 \leq \beta < 1$). However, the allocation is never decreased below what the neighbor is actually using, $r_i \geq \tau_i$. This algorithm is very similar to the approach taken for congestion avoidance, additive-increase/multiplicative-decrease [11]. Notice that we must give a min_alloc to each neighbor to allow for observation and therefore adaptation. The point of using weights instead of immediately adapting to the observation is to be able to capture the steady-state of the network without transients causing significant changes to the network.

After assigning capacities, we apply min-max fairness, min_max_fairness. The schedule is then created using the new assigned capacities, make_schedule. The schedule is then sent to all of the neighbors, and then the adaptation algorithm waits until it is time to update the schedule again.

Queue-based Adaptation

For the queue-based adaptation, every transmitted packet has an additional field indicating the number of packets pending in the transmitter's queue for the given receiver. The receiver uses the algorithm below to perform adaptation.

```
while (True):
  for i in neighbor_transmitters:
    packets_pending = last_packet[i].queue_size
    alloc = min(packets_pending/update_freq,1)
    r_i = max(α*alloc, min_alloc)
  t = min_max_fairness(r,M)
  sched = make_schedule(t, update_freq)
  synchronize_schedule_with_neighbors(sched)
  wait(update_freq)
```

The algorithm states that for every transmitter among the node's neighbor set, determine the number of packets still in the queue by examining the header of the last packet received from that neighbor, and then calculate the allocation that this value represents and weight it by α. Note that there is still a min_alloc given to each node. The rest of the algorithm follows the traffic-based algorithm.

While not explored, there exists a hybrid of traffic-based and queue-based adaptation that could be used to adapt around schedule conflicts. Assume that a node was given m slots and it is known that there are at least l packets left before the new schedule is started. If $l < m$, then it should be the case that at least l packets will be received if there are no schedule conflicts. If anything less than l is received and $l < m$, then it safe to assume this was caused due to a schedule conflict. In this case, reordering the slots in the schedule could

reduce the probability of a conflict. It is not clear as to how often this occurs and how much of an effect that this schema will have without cooperation and is left for future work.

5.5 Evaluation Model

To evaluate Listen-only Scheduling, I created a custom simulator that measures the performance, in terms of throughput and latency, of the network as a function of topology and load.

Simulator

The simulator follows the same system model presented in Chapter 2. The simulator takes as an input a network topology represented as a graph. A node consists of an independent transmitter and receiver and an output queue for every neighboring receiver, Figure 2.1.

All nodes are homogeneous with only different random seeds varying the initial state of the node. Edges in the graph represent the possible transmission channels that may exist. Transmission channels, for this study, are all treated as having the same throughput, one packet per slot. Interference is ignored because we assume that the modulation and coding scheme are set to be such that the amount of SINR is always achievable independent of interference.

Within consecutive schedule conflict situations within the same set of receivers, the transmitter will follow round-robin arbitration among the queues that have packets and whose corresponding receiver currently allows the node to transmit, determined by the neighbor schedules.

Execution of the simulator follows the synchronous discrete-time driven semantics where each discrete time step (one slot) is a slot where a packet may or may not be transmitted. It takes one slot for a packet to be transmitted from a transmitter to a receiver. Queues are of infinite size.

Packets are generated according to a Pareto On/Off distribution in order to mimic bursty traffic, Section 2.4. Each node is seeded with a different random seed for packet generation. However, both ON and OFF are the same for all nodes. For all experiments, ϕ and OFF are fixed. ON is varied to vary the traffic demand.

Because routing is not the focus of this paper, each node uses predetermined routing based on the shortest number of hops from source to destination. In the case of multiple shortest paths, a node transmits to the neighbor that will allow it to send the packet fastest.

Overheads such as switching cost and listen-only schedule packet overhead are not simulated due to the fact that the preambles of 60 GHz typically are the equivalent of 1000s of bits long, and the overheads we anticipate for any of our schema to be equivalent to the order of 10s of bits. As an example, 802.11ad has two possible preamble lengths, 3328 and 7752 symbols long [30].

Topology

For our experiments, we conducted simulation of two network topologies: first, a large random 1000 node network topology; second, a network topology that we considered possible in a real deployment.

The first topology consists of a 1000 node network where each node is randomly placed in a unit square with a threshold used to decided whether there exists an edge between two nodes or not. The average number of neighbors for the topology generated was 4.2 with a maximum of 14 neighbors and a minimum of one neighbor. The topology is shown in Figure 5.6.

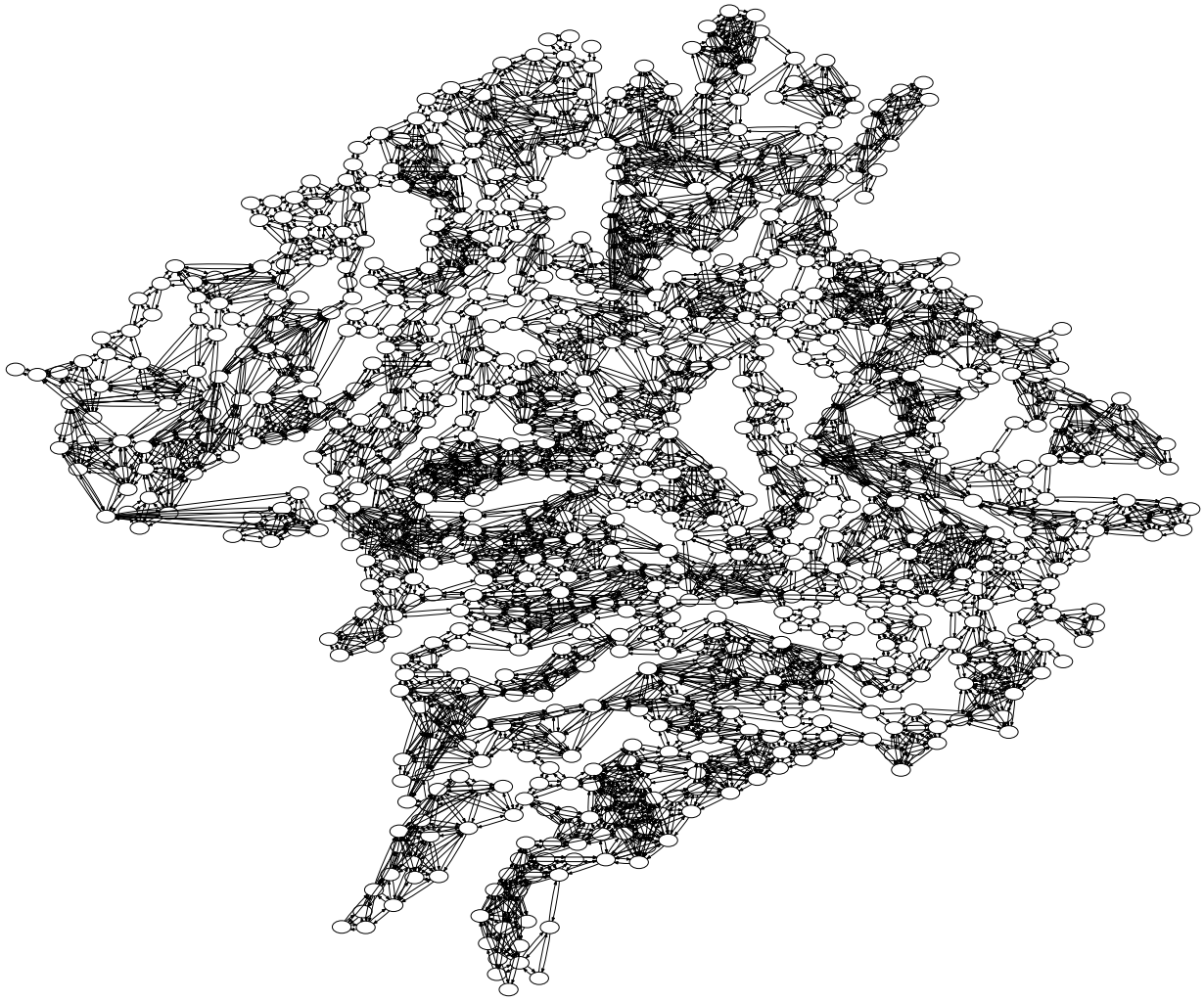


Figure 5.6: Random Network Topology

The focus of our work is to be able to eventually implement Listen-only Scheduling on real hardware. Therefore, a feasible topology was derived from the geometry of one of our

planned deployments, the BWRC, Figure 5.7. Nodes are assigned to every desk and edges are added between nodes that are within two meters of each other. In addition to these nodes, infrastructural nodes are also assigned to clusters of desks, and edges are added these nodes if they are within ten meters of each other. Infrastructural nodes represent higher power nodes capable of transmitting at the same rate as the desk nodes at a longer distance.

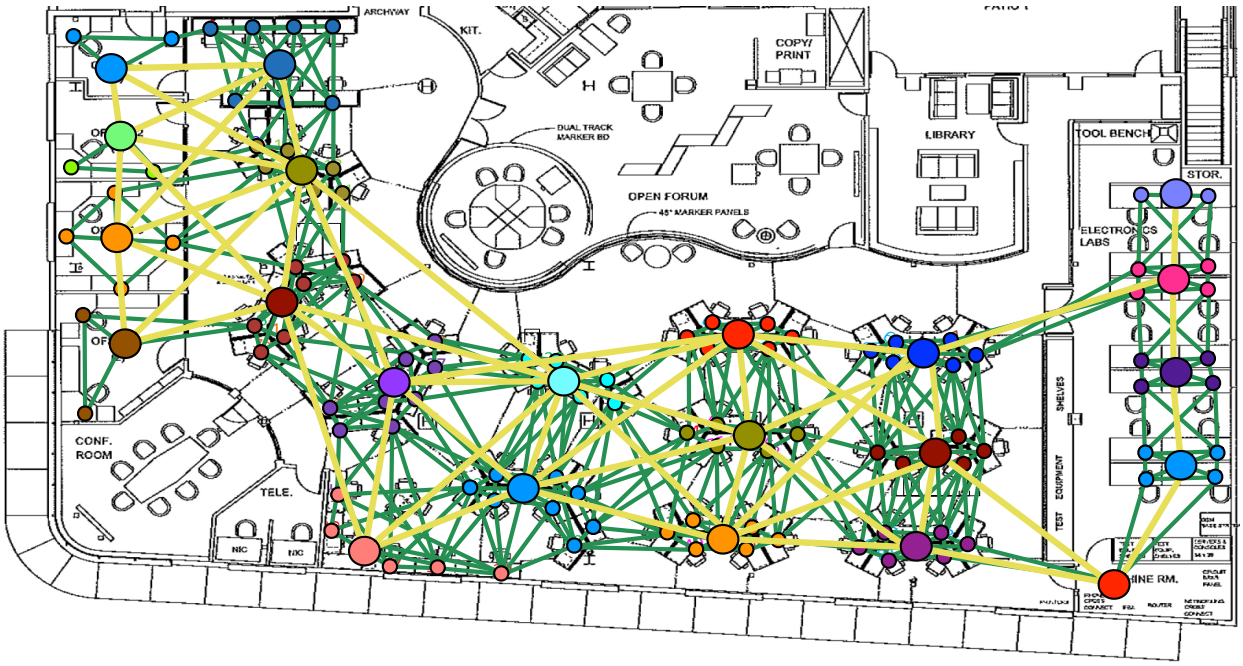


Figure 5.7: Proposed BWRC Network Topology

Traffic Models

Two different traffic models are evaluated: all-to-one and fixed-pair. All-to-one represent the typical LAN environment where all traffic goes to a gateway or server. For the random topology, the destination node is chosen randomly. For the BWRC, this is the machine room (bottom right node in Figure 5.7), which serves as both gateway and server. Important to note is that because of the node model the theoretical maximum throughput for the network is one packet per slot.

Fixed-pair communication is meant to represent what might happen in peer-to-peer communication. Before the start of the simulation, nodes are randomly paired up, uniformly. During the simulation, nodes communicate with their pairs. Note that routing is still predetermined, so the network will be subject to bottlenecks created by some intermediate nodes.

Metrics

As mentioned earlier, we are concerned with network throughput and latency. Adaptive Listen-only Scheduling is compared against a 'switch-every-slot' scheduler, 'perfect' adaptation scheduler, and directional slotted ALOHA (DSA).

The 'switch-every-slot' scheduler represents Listen-only Scheduling without adaptation where a node's schedule consists of one slot per neighbor. This scheduler serves as a baseline for the other algorithms.

The 'perfect' adaption scheduler is a scheduler that has a priori knowledge of the demand of its neighbors. Specifically, the scheduler knows if a packet is pending for the receiver. If a packet is pending and there are no other packets pending from other neighbors, then the packet is transmitted between the neighbor's transmitter and the scheduler's receiver. If there are multiple packets pending for different transmitter neighbors, then round robin arbitration is used to select which packet is transmitted. Note that the perfect adaptation scheduler makes only local decisions local to the node. Also note that there are no unexploited communication opportunities due to scheduling.

Directional Slotted ALOHA (DSA) represents random-access used instead of predetermined schedules. In DSA, receivers use omnidirectional receiving as described in Section 5.2, which allows determination of direction of the transmitter. It is assumed in this strategy that neighboring transmitters have knowledge of which direction to use for a given receiver. Neighboring transmitters send packets without sensing as there is no opportunity to sense due to deafness or possible issues of exposed node. If two neighboring transmitters transmit to the same receiver in the same slot, then this is considered a collision and neither node is able to successfully communicate during that time slot. If there is no collision, an acknowledgement is transmitted to the originating node near the end of the slot. A neighbor who transmitted waits for this acknowledgement at the end of its transmission. If no acknowledgement is received, the neighbor stores in its table for that neighbor a back off time determined using random exponential backoff. Note that the acknowledgement takes up transmit time and is not represented in the receive schedule except as part of a small overhead in the next transmit packet.

5.6 Results

The performance of Listen-only Scheduling was evaluated for twelve different random seeds and run for 10,000,000 slots. The random seed varies the packet generation as well as the offset for schedule updates to simulate schedule updates not happening simultaneously. The schedules consist of only 'One' slots. There are no slots of 'None' or 'Any' due to the fact that discovery is not being simulated. The order of the slots are randomly permuted at each schedule update. The length of a schedule, M, is set to 100. For the traffic-based adaptation algorithm, the growth α was selected to be 1.25, β as 0.5, and γ as 1.25. For the queue-based, the weight α was selected to be 0.8. These values were selected based on a parameter space

exploration.

All-to-One

The all-to-one traffic model results for the BWRC topology for the different seeds are shown in Figures 5.8 and 5.9. The average latency of all the nodes is plotted against the received throughput at the machine room node using log(base 2)-linear scale.

As can be seen in the figures, 'switch-every-slot' saturates at around 20% utilization and has strictly worse performance than any other other algorithm. For all random seeds, DSA saturates around 60% utilization. However, depending on the seed, its performance is better than the adaptation algorithms up until 20% to about 39% utilization. The mode value being 33% utilization. The adaptation algorithms performs well as compared to the perfect scheduler. After 50% utilization, the adaptation algorithms are always within 2x of the perfect and within 4x before it. Also, for this traffic model, the traffic-based adaptation has a slight advantage against the queue-based algorithm up until 70%. Both adaptation algorithms and the perfect saturate at approximately the same point, which is very near the theoretical maximum of one packet/slot.

The all-to-one traffic model results for the 1000 node random network topology for two different seeds are show in Figure 5.10 that are representative of the random seeds. Like the BWRC topology, we see the same saturation points and the similar crossover points for the adaptation algorithms and DSA. The biggest difference is the gap between the two adaptation algorithms and the perfect scheduler. While in the BWRC topology the gap between the adaptation algorithms was very small (typically within 10–40% of each other), the gap between the traffic-based and queue-based in the random network is larger (typically between 50–90% of each other). The gap between the perfect and the adaptation algorithm is also consistently less than 4x. Lastly, the crossover point for the queue-based adaptation with better performance than the traffic-based is 80% utilization instead of 70% as seen in the BWRC topology.

Fixed-pair

For the fixed-pair communication, the results for the BWRC are shown in Figures 5.11 and 5.12. There was a larger variation in the results for the fixed-pair communication compared to the all-to-one due to the fact that the random pairing can drastically change the performance of the network. In addition, for the all-to-one communication, congestion-avoidance algorithms would have less effect on the network due to all traffic eventually going to the same place. However, for fixed-pair communication, the network could potentially benefit greatly if congestion-avoidance was used.

The queue-based algorithm consistently showed better capacity and latency than the traffic-based algorithm with the exception of very low utilization where traffic-based performed slightly better. DSA showed better performance than the queue-based Listen-only Scheduling up until 15–30% of the utilization with a mode value of approximately 20%. As

far as performance, the queue-based adaptation algorithm performs within 2x of the perfect algorithm up until it starts to saturate. DSA consistently saturated around 50–60% similar to the all-to-one traffic model. The queue-based adaptation algorithm was not able to achieve the same capacity as the perfect but was always within 20% of the perfect's capacity.

Once again, for the random topology, we see the same crossover and saturation points compared to DSA and the perfect scheduler. The largest difference in the random topology is the fact that the traffic-based adaptation has higher performance at higher utilizations than the queue-based adaptation. This crossover point is roughly at 70% utilization, comparable performance to the all-to-one traffic model.

Maximum Throughput

To evaluate the maximum achievable throughput for the listen-only scheduling approach without explicitly handling conflicts, we use the same scenarios as before except we set the nodes to send a packet every slot, and we fix the routing to be deterministic instead of allowing opportunities to send to neighbors of equal distance away. For the fixed pair scenario, it is necessary to evaluate multiple different pairings. For this, 500 random seeds are used, and we also give the relative error for a 95% confidence interval.

For the all-to-one traffic model, the maximum throughput can be determined for all traffic models simply by noting the fact that the bottleneck is the destination node and therefore can achieve a maximum throughput of one. For the fixed-pair traffic model, the maximum throughput must be determined on a per seed basis.

I calculate two maximums for the fixed-pair traffic model. The first, *Max Global*, is the maximum throughput through the network calculated using a linear program that solves the maximum multi-commodity flow problem with a fixed set of routes. This maximum throughput does not take into account local node fairness. The second, *Max Local*, takes into account the local fairness due to round robin arbitration at each node as well. The details of these algorithms can be found in Appendix D.

For the all-to-one traffic model, all of the algorithms, except DSA, were able to achieve the maximum throughput of one packet per slot for both topologies. This is understandable as it is only necessary to saturate the destination node to achieve maximum throughput. Because of the high contention for throughput, the throughput of DSA was effectively zero.

The fixed-pair traffic model relative performance results compared to the two maximums for the random and BWRC topology are shown in Tables 5.3 and 5.2, respectively.

For the random topology, listen-only scheduling with the perfect scheduler performed within 73.1% of the maximum throughput considering round robin arbitration. However, it performed at 35.6% compared against the maximum global throughput. For the BWRC, we can see that Listen-only scheduling can achieve 82.2% relative performance. 22.2% relative performance is lost due to round-robin arbitration. We believe that the losses of 26.9% and 17.8% in both topologies compare to *Max Local* are due to schedule conflicts.

Table 5.2: Maximum Throughput for Fixed-Pair — Random

	Max Local		Max Global	
	Rel. Performance	Rel. Error	Rel. Performance	Rel. Error
Perfect	72.7%	0.4%	35.4%	0.6%
Queue-based	40.0%	0.9%	19.5%	1.1%
Traffic-based	43.2%	0.8%	21.1%	1.1%
DSA	22.6%	2.4%	11.1%	2.7%
Switch-every-slot	14.5%	0.5%	7.1%	0.4%

Table 5.3: Maximum Throughput for Fixed-Pair — BWRC

	Max Local		Max Global	
	Rel. Performance	Rel. Error	Rel. Performance	Rel. Error
Perfect	82.2%	0.4%	60.0%	0.8%
Queue-based	60.4%	0.8%	44.1%	1.1%
Traffic-based	52.7%	1.2%	38.7%	1.6%
DSA	30.1%	3.2%	22.2%	3.5%
Switch-every-slot	23.3%	0.9%	17.0%	0.8%

Note that our traffic generator does not have feedback such as what occurs in TCP. Therefore, the natural backoff of links that would normally occur is the main source of why our simulation results do not result in good results relative to the *Max Global*. In future work, it would be interesting to see the relative performance to *Max Global* with TCP like traffic.

5.7 Discussion and Future Work

Based on the results, it can be seen that the adaptation algorithms perform very well (within 2x of perfect) at high load and operates reasonably well (within 4x of perfect) during very low load, independent of listen-only schedule conflict avoidance. In fact, the largest source of latency during the low load is most likely due to the ordering of the slots as most of the slots are in fact unexploited. It is during very low load that it would make sense to use random-access (such as DSA) instead of a schedule-based approach such as the hybrid approaches taken in 802.11. Random-access does not make sense at high load as it can severely limit the total throughput of the network, 60%, while Listen-only Scheduling with adaptation can achieve 80–95% of the total potential throughput.

In the system described so far, every node has had a common notion of a slot. Because a listen-only schedule is local to a given node, there is no explicit reason to maintain this assumption. If every node were to keep time synchronization information for each of its

neighbors, then global synchronization would be unnecessary. This is a significant advantage to global synchronization which limits overall scaling, especially true for indoor networks.

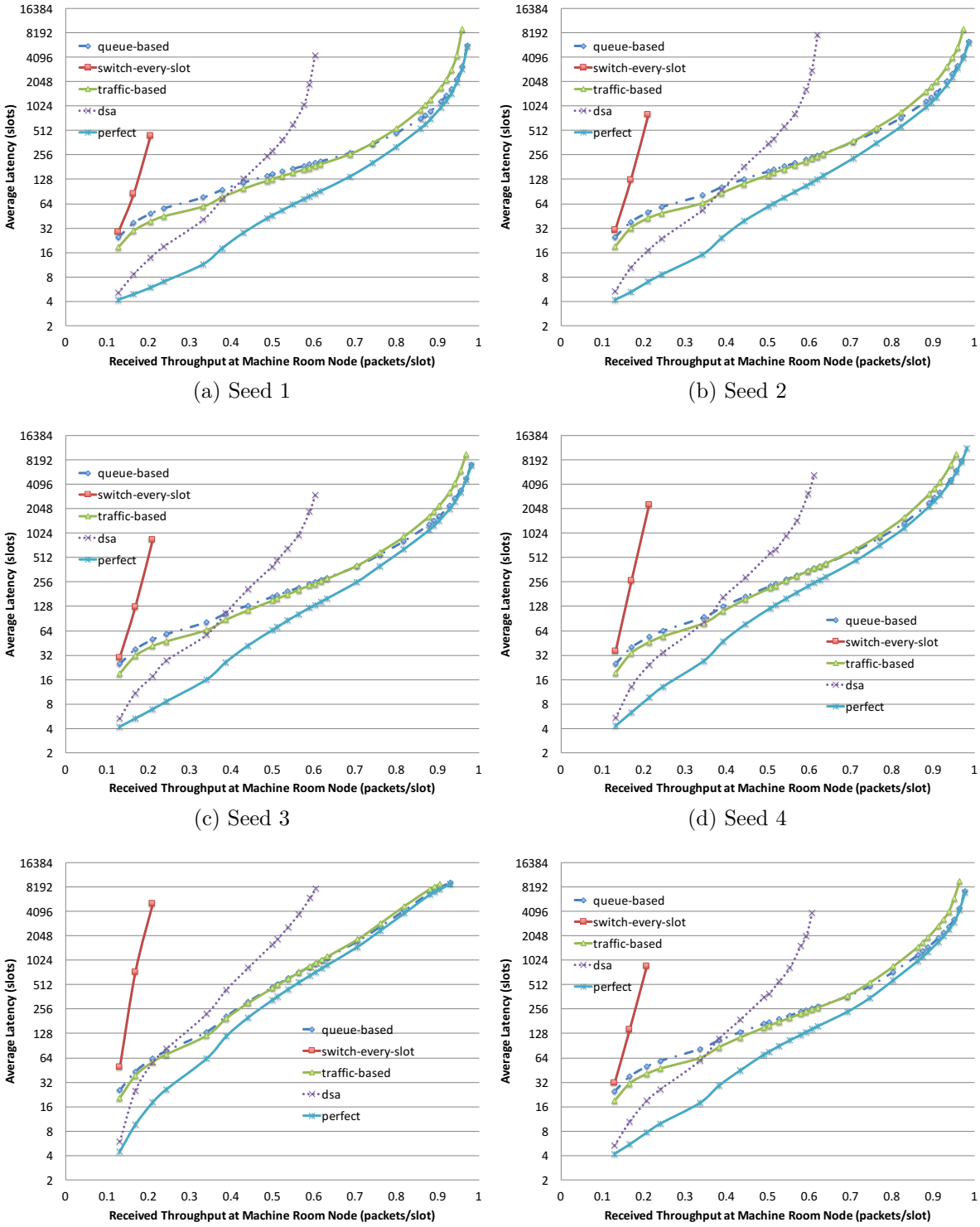
The assumption of static node positions and directions can be relaxed to support mobility. If nodes do not move very quickly relative to the distance between them, the new direction will not be too distant from the previous direction used. If one were to leverage this in combination with a sufficient frequency of communication between two nodes, then it is possible to continuously update the directions of the nodes. If mobility is very high and unpredictable, then the only possibility would be to learn the direction of each node on every slot, increasing overhead.

As mentioned earlier, the preambles that are present at 60 GHz could be fairly significant. Therefore, the size of the slot will need to be carefully considered to minimize the overhead of the preamble. Depending on the traffic model, multislot frames may be necessary to minimize the preamble overhead. As a consequence, individual 'One' slots for the same neighbor will be need to be scheduled explicitly together increasing the amount of history necessary to maintain fairness.

There are still many open problems in Listen-only Scheduling itself. Using the 'perfect' scheduler 20–30% of the maximum throughput is lost due to inefficiencies of local scheduling. It should be possible to obtain even better scaling if nodes are allowed to coordinate with each other to minimize latency and jitter. This approach may make the hybrid of traffic-based and queue-based, as mentioned in Section 5.4, more attractive. It would also be good to evaluate the throughput of the network in a more realistic setting where the throughput of the links are not homogeneous and instead vary with distance and take into account possible interference. Listen-only Scheduling should also be studied in the context of TCP/IP traffic flows. The effect of Listen-only Scheduling on routing should also be studied.

5.8 Summary

In summary, I have presented a link management scheme called Listen-only Scheduling that allows for completely distributed control of directional antennas in wireless networks. Even without coordination, Listen-only Scheduling performs well when used with adaptation algorithms and consistently has higher performance in terms of latency and achieves better throughput scaling than DSA. I also showed that in spite of the new inefficiencies, schedule conflicts, introduced, Listen-only Scheduling performs within 20–30% of theoretical maximum throughput. With limited coordination among nodes, I believe the gap can be closed even further of the theoretical.



(a) Seed 1

(b) Seed 2

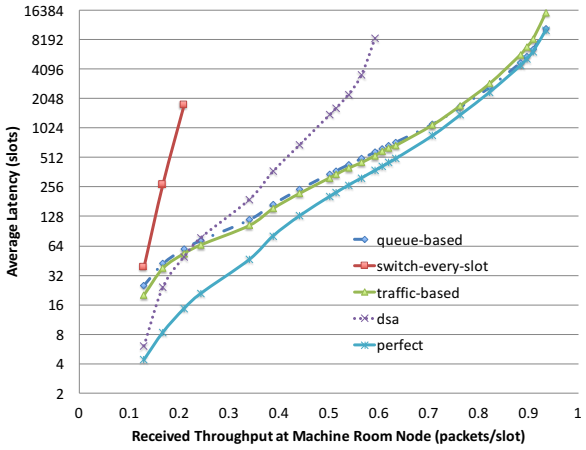
(c) Seed 3

(d) Seed 4

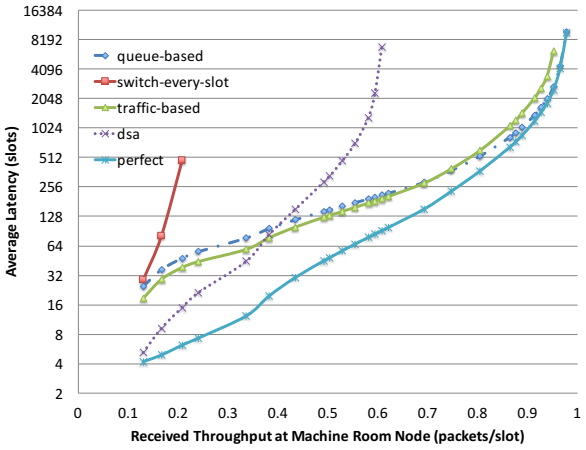
(e) Seed 5

(f) Seed 6

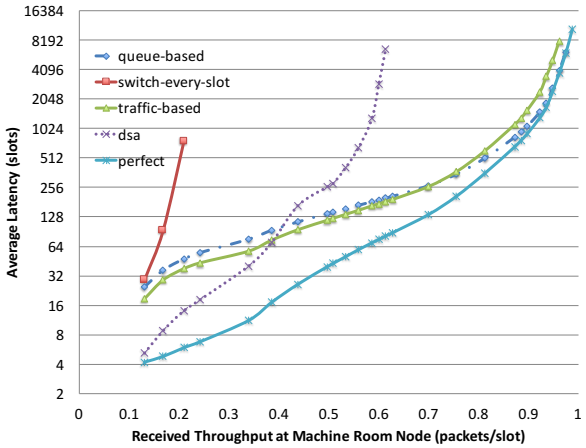
Figure 5.8: All-to-one Traffic Model: BWRC Topology



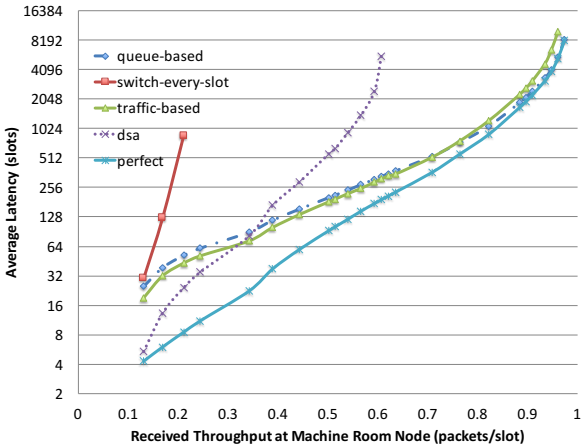
(a) Seed 7



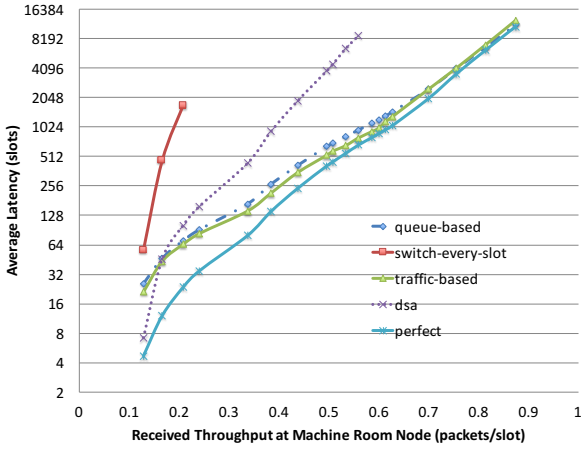
(b) Seed 8



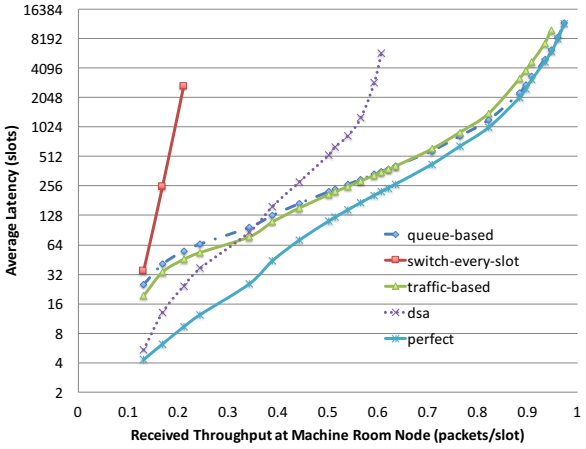
(c) Seed 9



(d) Seed 10



(e) Seed 11



(f) Seed 12

Figure 5.9: All-to-one Traffic Model: BWRC Topology

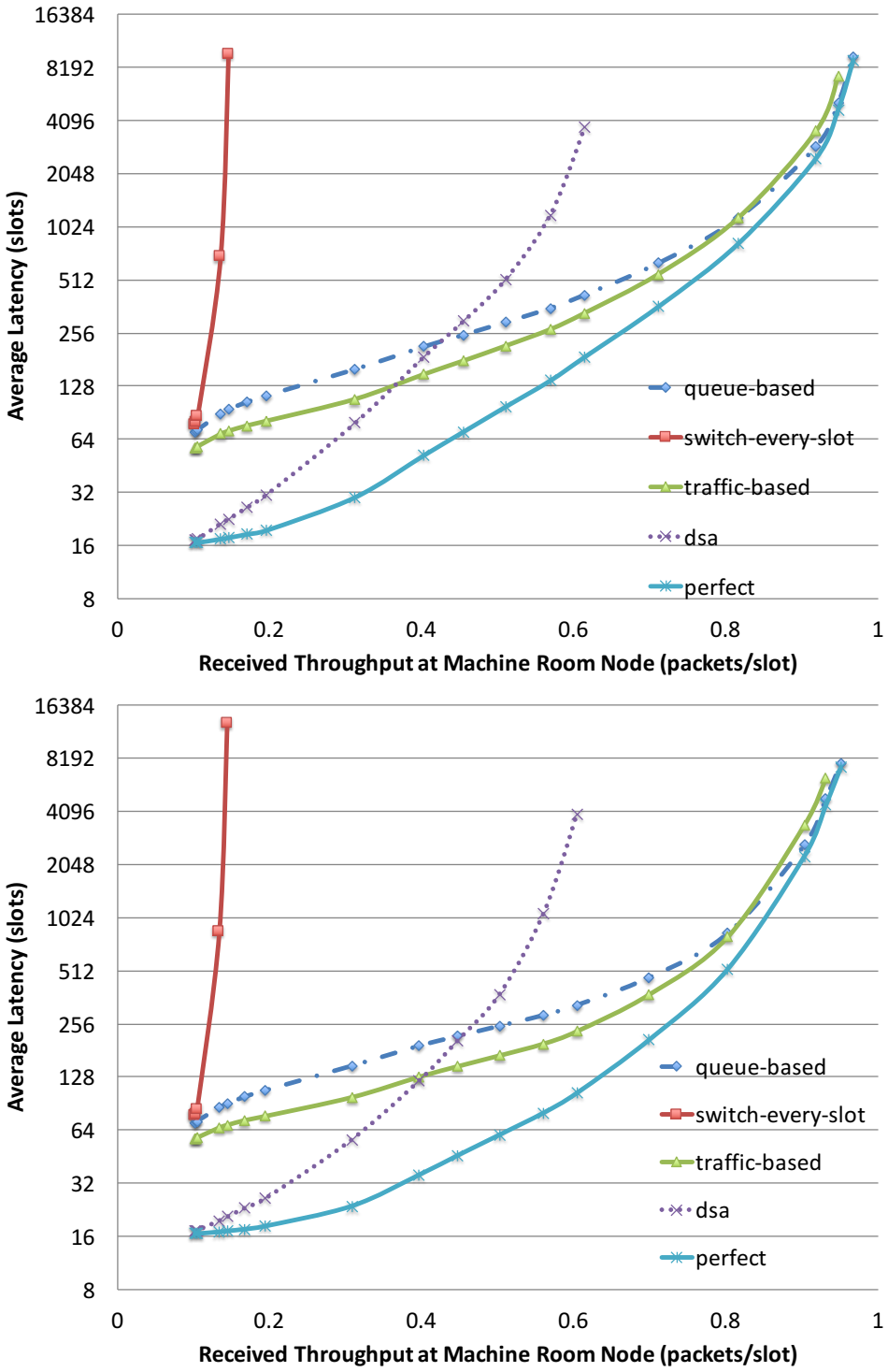
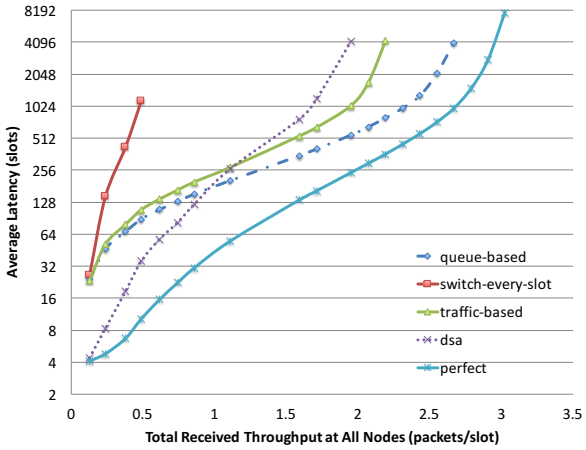
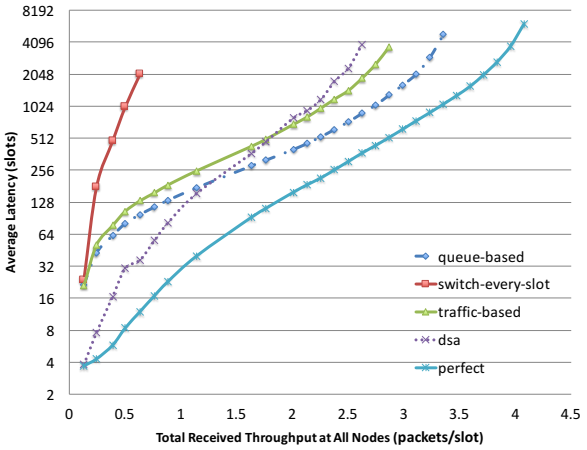


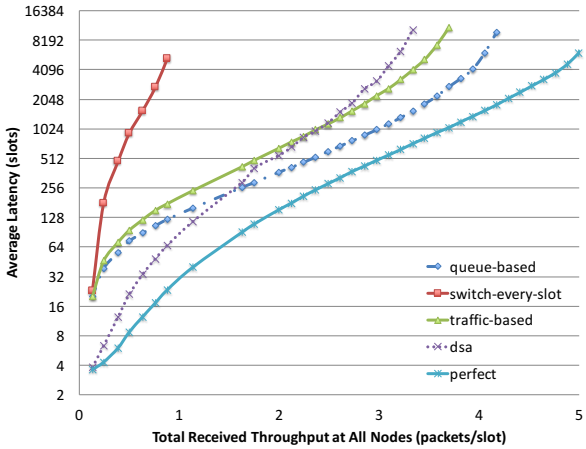
Figure 5.10: All-to-one Traffic Model: Random Topology



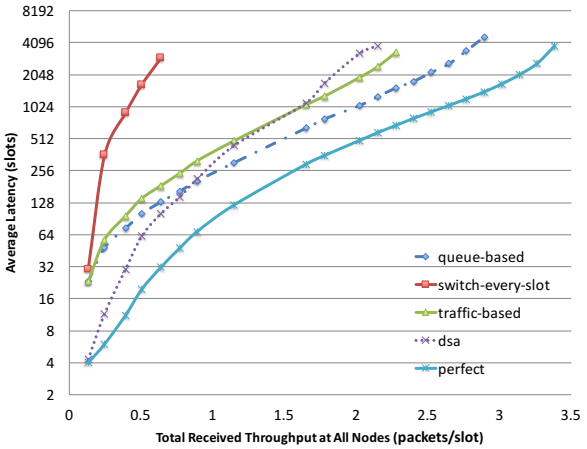
(a) Seed 1



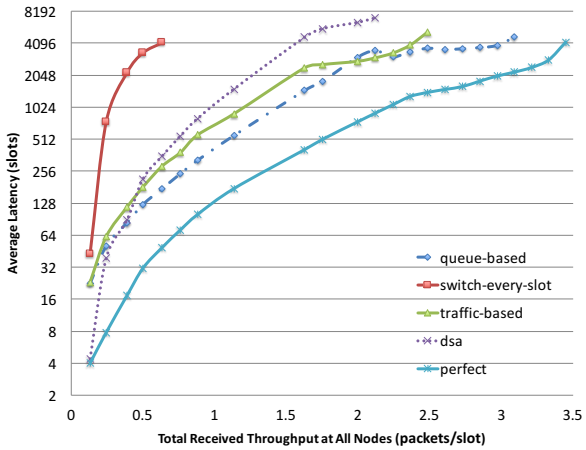
(b) Seed 2



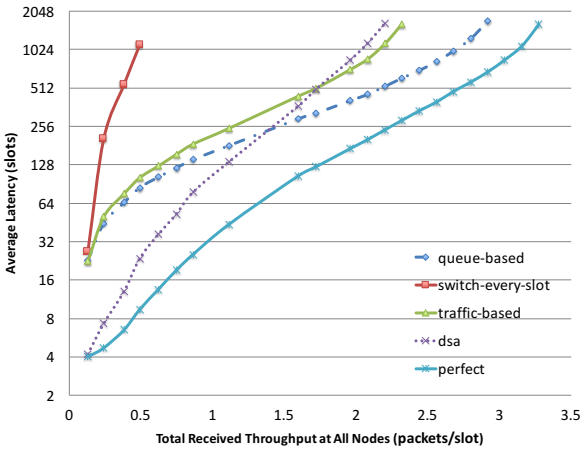
(c) Seed 3



(d) Seed 4

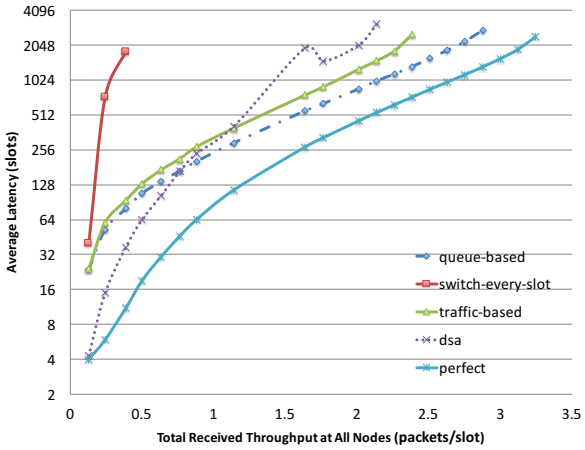


(e) Seed 5

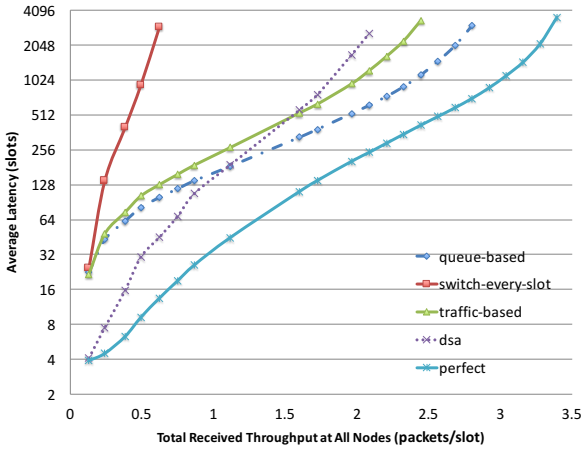


(f) Seed 6

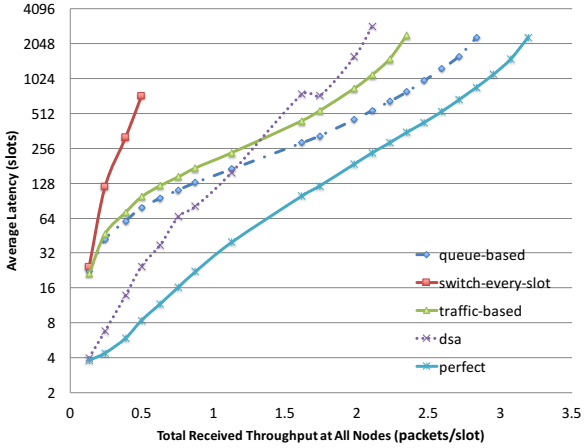
Figure 5.11: Fixed-Pair Traffic Model: BWRC Topology



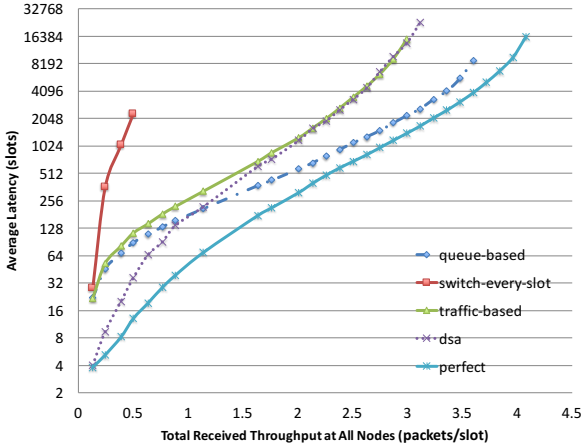
(a) Seed 7



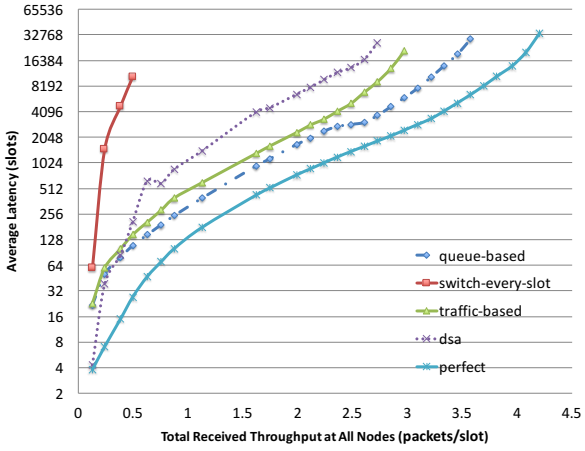
(b) Seed 8



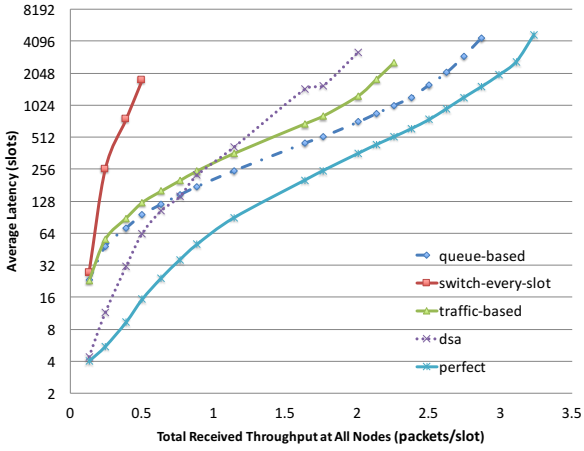
(c) Seed 9



(d) Seed 10



(e) Seed 11



(f) Seed 12

Figure 5.12: Fixed-Pair Traffic Model: BWRC Topology

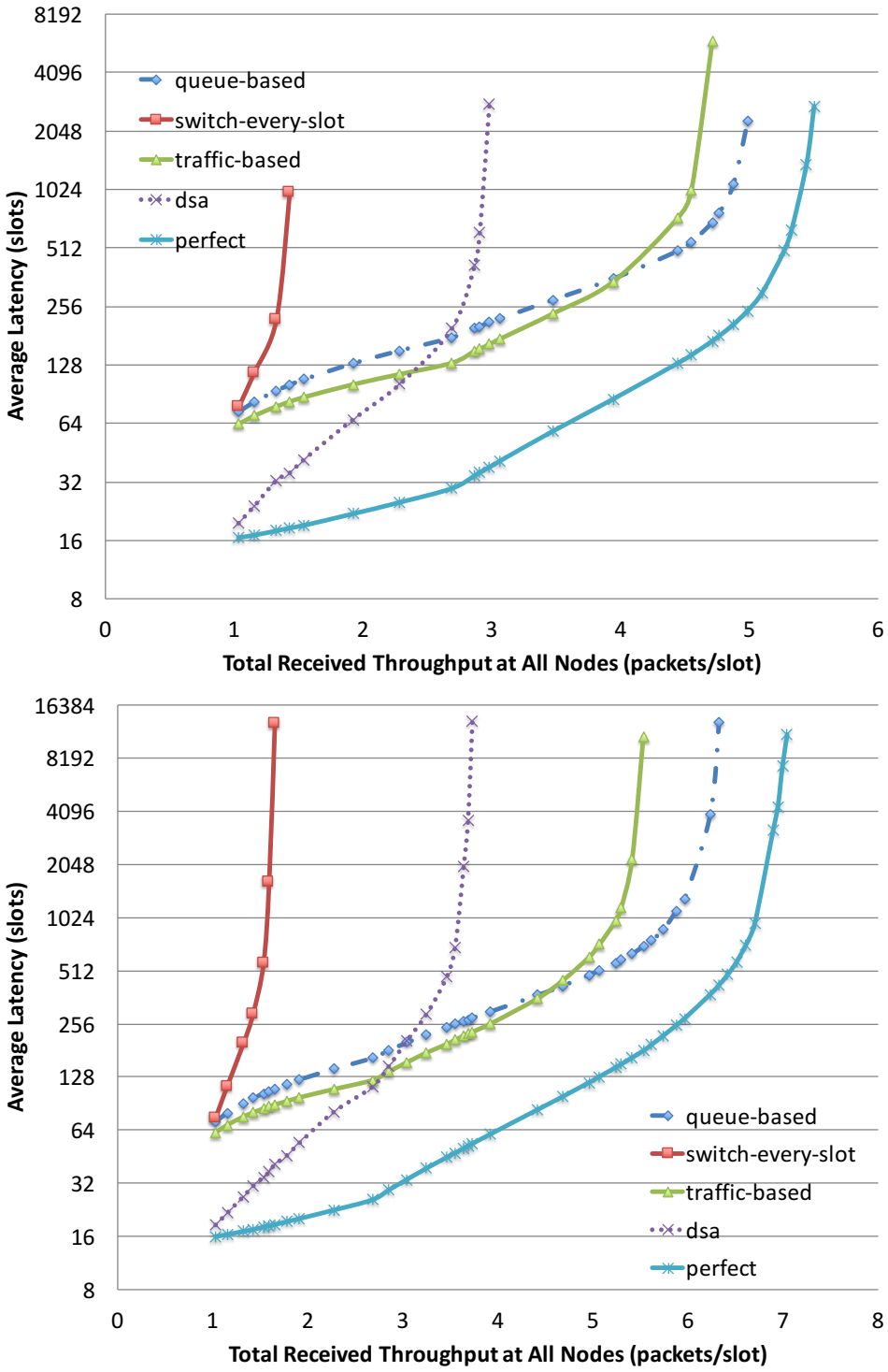


Figure 5.13: Fixed-Pair Traffic Model: Random Topology

Chapter 6

Future Directions and Conclusions

This final chapter presents new areas worth exploring particularly in cross-layer optimization. In addition, some preliminary work in interference aware routing is presented. The thesis is drawn to a close with conclusions noting some of the high-level takeaways.

6.1 Different Antenna Array Geometries

This work mainly addressed rectangular arrays with $\lambda/2$ spacing. However, this is obviously not the only possible antenna array geometry. More antenna geometries should be explored in terms of their effect on interference and in turn the capacity scaling. In particular, there are commercial antenna arrays (such as in Figure 1.1a) where the spacing of the antenna array is not $\lambda/2$ but is instead a larger spacing. Preliminary work suggests that rectangular arrays with spacings up to λ cause less interference.

An alternative approach may be to take something like Peter Bevelacqua's dissertation [8]. Given a probability distribution function (such as in Chapter 4), solve for the antenna array geometries such that it minimizes interference using the properties common at mmWave and above (e.g. 10s-100s of antenna elements and planar arrays). Using these results, one could then simulate the overall interference and determine the scaling properties.

6.2 Interference Aware MAC

At the medium access control (MAC) level, it is likely possible that one should be able to mitigate interference by using time-division multiplexing with interference-aware scheduling. My previous work has focused on how to efficiently coordinate the schedule of when nodes communicate using a distributed schedule approach.

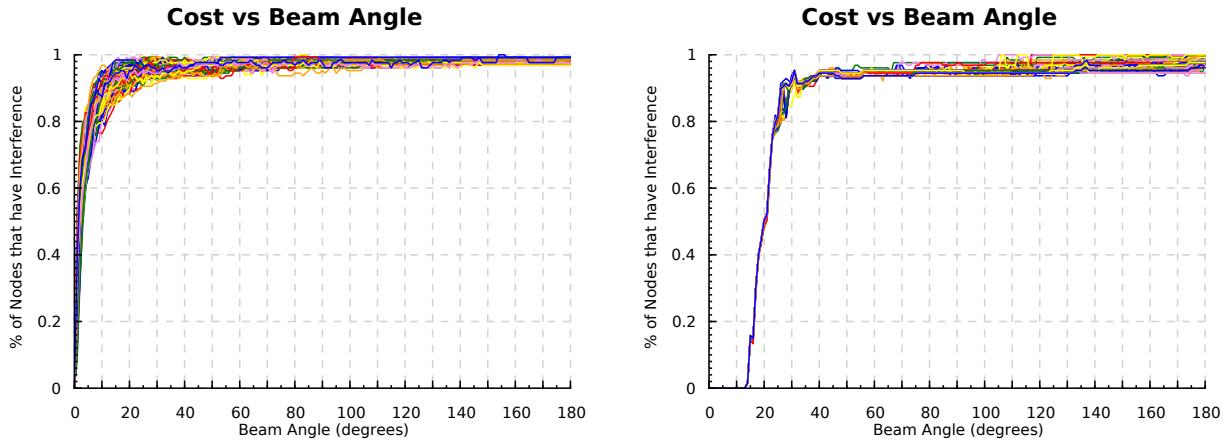
From the point of view of interference, some combination of links are more orthogonal than others. Therefore, if nodes could observe what links are causing interference, than it should be possible to select a time schedule such that the capacity of the network is maximized. There are three main factors that makes this problem difficult: two links being truly

orthogonal is rare in even medium density networks; establishing the links responsible for interference can be time-consuming or difficult; and the optimal solution needs information about the entire network leading to potentially significant communication overhead.

6.3 Interference Aware Routing

As with the MAC-level, interference avoidance techniques can also be applied at the network level using scheduling, except in this case the concern is scheduling the paths through the network instead of scheduling the operation of individual links. Preliminary work has shown that interference can be avoided by leveraging short distance links similar to the Gupta-Kumar result [26]. However, for all but the most dense of networks, it is likely that better capacity than Gupta-Kumar can be achieved by leveraging the directionality of links.

In preliminary work, very simple models of beam were used, either being in the mainlobe or not being in mainlobe. Using the same BWRC geometry as seen in Figure 3.5, all paths are calculated from every node to every other node. The shortest path using Dijkstra's shortest path algorithm is calculated using two different methods of assigning weights: hop count (Figure 6.1a) and interference (Figure 6.1b). Interference is calculated assuming that the receivers are omnidirectional and they are all on. Reflections are not simulated. The particular direction chosen for the beam is based on what minimizes the interference. Figure 6.1 displays a line for each path, and since mainlobe or not is being used, we can show the percentage of nodes that experience interference as a function of beam angle.



(a) Hop Count Weights

(b) Interference Weights

Figure 6.1: Interference-Aware Routing: Interfered Paths

As can be seen, if one uses hop count as a method for routing, nearly all nodes experience interference for all beam angles. However, when interference is taken into account, it is possible to eliminate interference for beam angles up to 20 degrees. This is preliminary work, so more realistic antenna array patterns have not been studied yet. However, the

results are promising, suggesting that it is possible to reduce the amount of interference in the network, achieving greater network capacity.

Effects on Paths Chosen

Another preliminary experiment was performed examining how beam width would effect the routing. The same model is used as stated earlier (i.e. of being in the beam or not being in the beam, the same BWRC geometry, interference calculation, and etc).

The following figures in this subsection have the following convention. Nodes are circles. Edges are lines. The source node is blue. A white node indicates that the node experiences no interference. An orange node indicates that the node is receiving interference. A white edge indicates that the edge is used for routing. A red edge (color is relatively faint) are edges that could have been chosen but were not. Though too small to read, the green numbers are the edge weights.

The baseline for this experiment is considering the paths chosen when the beam is infinitely narrow, Figure 6.2. The source node for all of these experiments is located in the upper left corner.

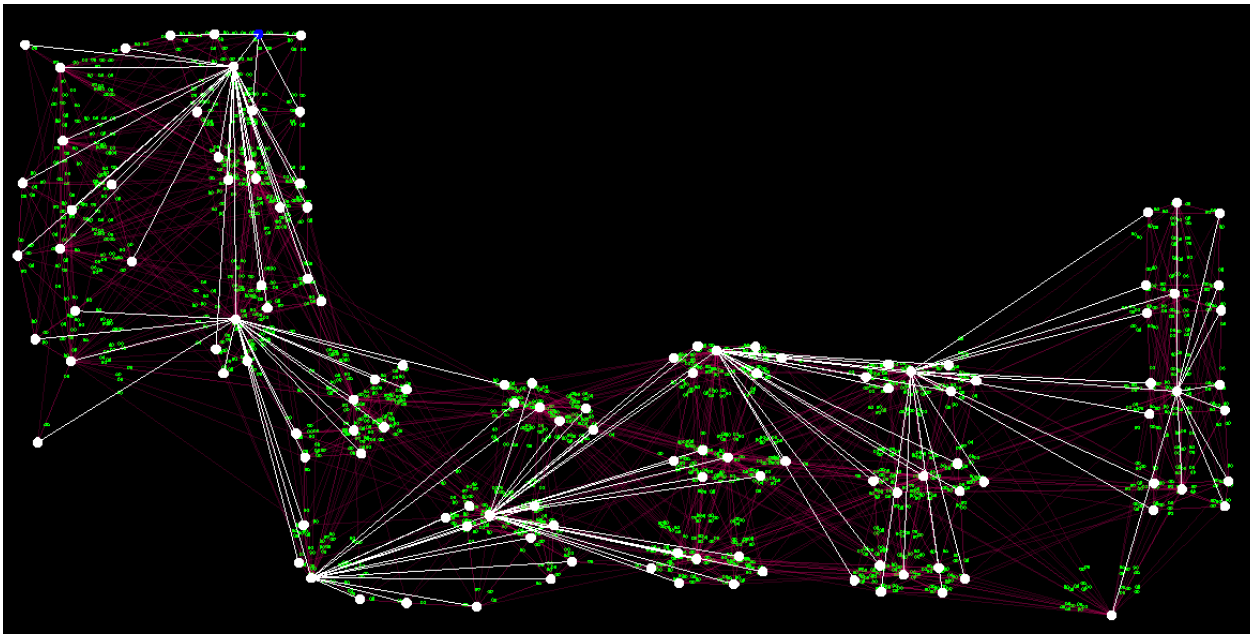


Figure 6.2: Paths of Interference-Aware Routing: Infinitely Narrow Beams

Figure 6.3 shows the routes for beam widths of 6°, 15°, and 45°. For the 6° beam width, Figure 6.3a, one should note the edge lengths are essentially the same as the baseline of infinitely narrow beams. However, the orientation of the lengths have changed. The reason the orientations have changed is due to the fact that if a node points its beam outwards, there is less of a possibility to interfere with other nodes. This corroborates with Section 4.2

that showed pointing inwards will maximize the number of nodes that one can communicate with. In this scenario, the routing is taking advantage of the contrapositive of this statement. If reflections were to be simulated, it is still likely that a similar route will be chosen because beams that reflect will be subject to reflective loss as well as increased path loss.

As the beam width increases, one should note that the edges chosen become shorter and shorter. However, up to 60° beam width, Figure 6.4a, there is still no nodes that are receiving interference. However, this comes at the tradeoff of increased hop count due to the use of shorter links. At 90°, Figure 6.4b, there are nodes that are experiencing interference (in orange). However, none of the interfered nodes are intermediate nodes. Therefore, only when communicating to specific nodes will the capacity of the path be reduced.

One should note that even at 120°, Figure 6.4c, a large portion of nodes are still without interference. In addition, three of the interfered nodes out of the 127 nodes are not in the leaves of routes. If the source node must communicate using an intermediate, the overall throughput may be reduced because of the intermediate nodes acting as bottlenecks.

6.4 Conclusions

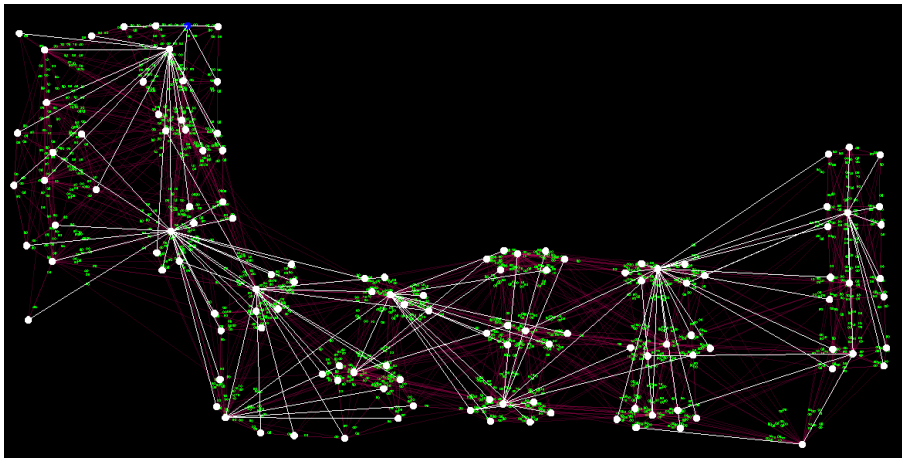
This work took a system-level viewpoint of how to design certain aspects of directional wireless mesh networks. One of the key ideas of this work was examining at how to the improve the network as a system as opposed to more traditional views of optimizing from a single standpoint. In addition to the cross-layer optimization that was proposed, I showed how if you consider antenna array geometries in a system, then you might design your antenna array differently (as a linear array) than if you were simply working from optimizing a beam pattern assuming that will be the best design point.

Another key idea that came from this work that was highlighted in both listen-only scheduling as well as the interference mitigation is that there exists algorithms for distributed control that perform well. Particularly, because receivers have information from the environment (either packets or electromagnetic waves), they are able to optimize for this. Clearly, cooperation would be useful. However, this implies that the baseline of no cooperation from nodes still has a sufficient amount of performance.

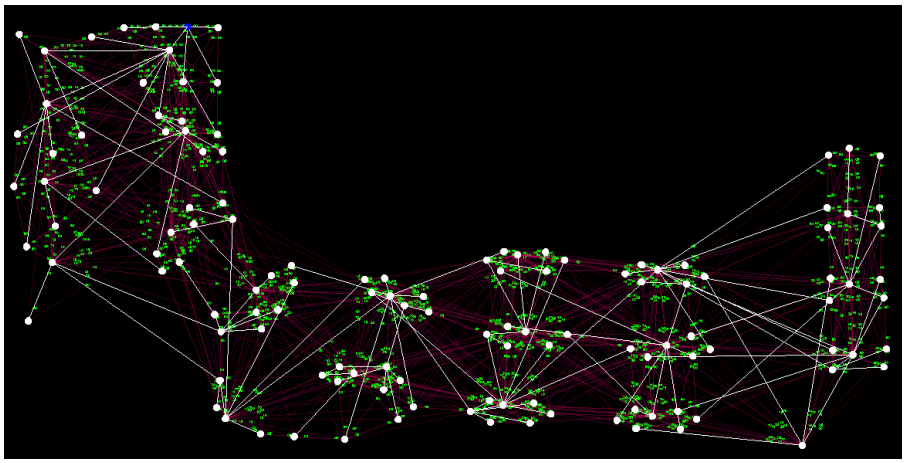
While I mainly looked at mmWave, a great deal of this work extends to other frequencies. Listen-only scheduling is a general technique that could be applied even at lower frequencies. The interference mitigation techniques demonstrated are agnostic to the antenna array itself, and for the higher antenna count simulations, it may be that they only exist above mmWave. This is also true for designing antenna array geometries.

Even though this work represents only a dent in the knowledge of directional wireless mesh networks, I believe that there are many doors left open for exploration. In particular, cross-layer optimization between physical, MAC, and network layers is relatively an open field. The preliminary work presented here are only a few ideas of many. In addition, this work, due to the lack of available hardware, relied on simulation and analysis to draw conclusions. There remains a significant amount of work as to how directional wireless mesh

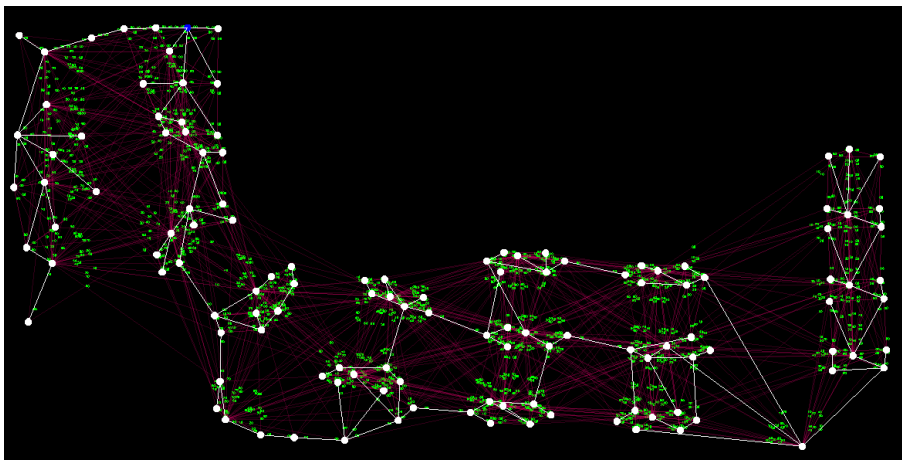
networks perform in real environments with real hardware and, with their limitations, should lead to new areas of exploration. Hopefully, this work opens more doors than it closed in this area and will serve to inform real physical systems.



(a) 6° Beam

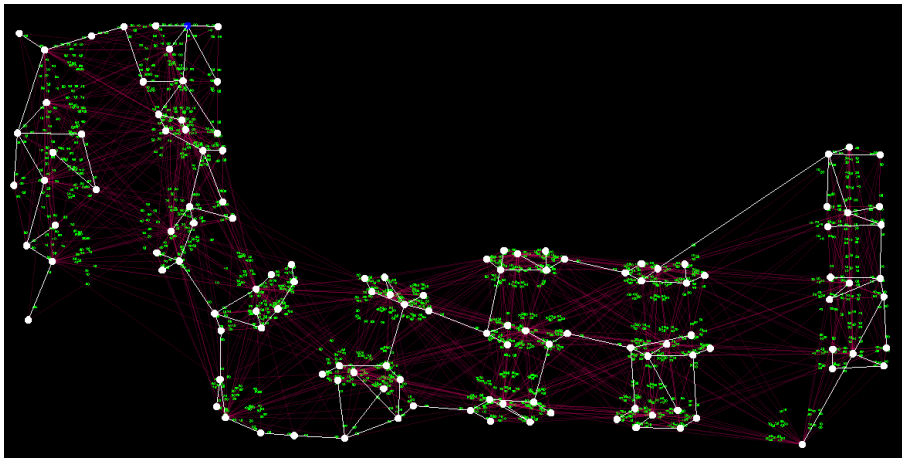


(b) 15° Beam

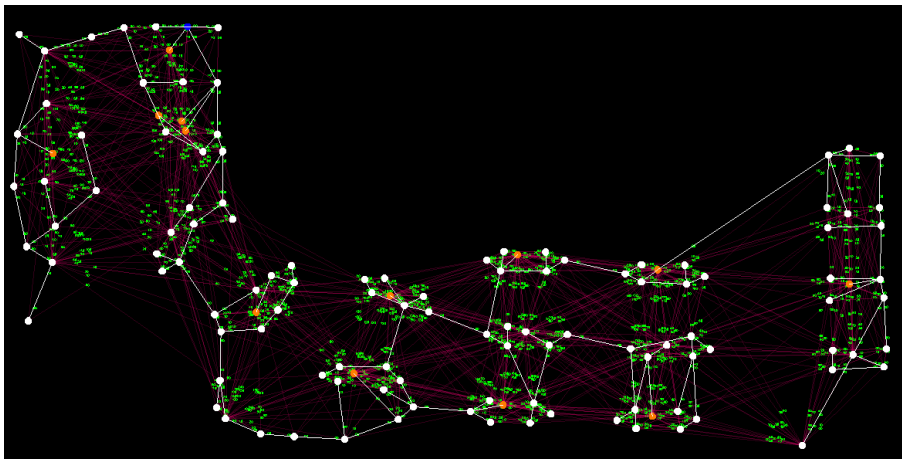


(c) 45° Beam

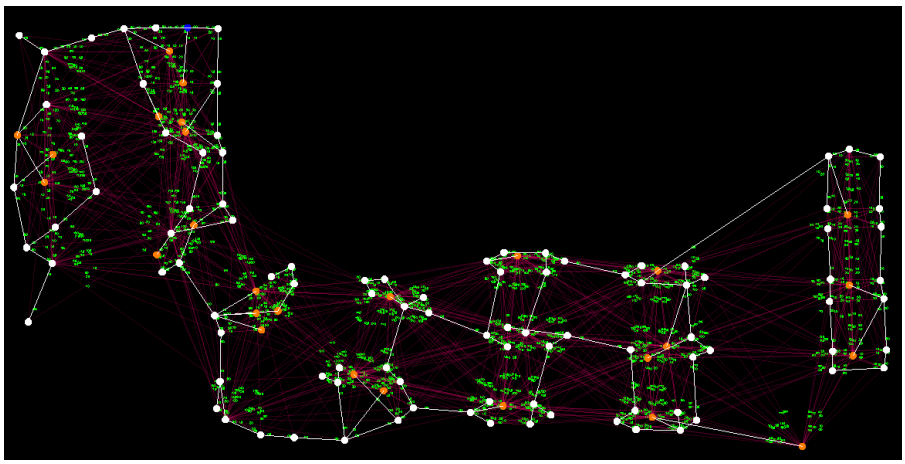
Figure 6.3: Paths of Interference-Aware Routing: 6–45° beam width



(a) 60° Beam



(b) 90° Beam



(c) 120° Beam

Figure 6.4: Paths of Interference-Aware Routing: 60–120° beam width

Appendix A

Interference Mitigation — Simulation Executor

The executor code below is given in Haskell. Note that the `BangPatterns` extension is necessary for the below code.

A.1 Libraries

The necessary libraries are given below.

```
import Control.Arrow(first)
import Control.Monad(when)
import Control.Monad.Trans.Class
import Control.Monad.Trans.RWS.Strict hiding (tell)
import Data.Functor.Identity(Identity, runIdentity)

import Data.HashSet(HashSet)
import qualified Data.HashSet as HS
import Data.HashMap.Strict(HashMap)
import qualified Data.HashMap.Strict as H
import Data.Sequence(Seq,(|>))
```

A.2 Types

The types used in the executor are given below.

Time is represented as a double floating point number.

```
type Time = Double
```

The transmit and receive nodes are represented by integers.

```
type TxNode = Int
type RxNode = Int
```

A message is either the beginning of the message or the end of the message and contains the time that the message occurs along with the source and destination node.

```
data Message = MsgBegins { time :: !StartTime, srcTx :: !TxNode, dstNode :: !RxNode }
             | MsgEnds   { time :: !EndTime,   srcTx :: !TxNode, dstNode :: !RxNode }
    deriving (Show,Eq)
```

`ActiveLinks` is the set of links (transmitter/receiver pairs) that are currently transmitting.

```
type ActiveLinks = HashSet (TxNode,RxNode)
```

`SolverFunc s` is the function that given a set of active links outputs a result, `s`. This function is one of the arguments to the executor.

```
type SolverFunc s = ActiveLinks → s
```

The executor generates a `SimulationTable s` that maps `ActiveLinks` to solutions from `SolverFunc s`.

```
type SimulationTable s = HashMap ActiveLinks s
```

`SolverState s` is the intermediate state of the executor. It stores the current links that are active and previous results of the solver function.

```
data SolverState s = SS { activeLinks :: HashSet (TxNode,RxNode),
                          usedBefore :: !(SimulationTable s) }
```

Note that based on these types that for a given `ActiveLinks` that the solution is always the same. Therefore, the astute reader will note that the executor must be memoryless. Therefore, `SimulationTable s` existing as part of `SolverState s` is for performance (amortization purposes only) and is not strictly necessary for functionality.

For convenience, the parameters of the executor are bundled together. The default value is used for when there are no links active.

```
data SolverParameters a = SParams { defaultValue :: a, solverFunc :: ActiveLinks → a }
```

The results of the simulation table as well as s time sequence with the result at each unique time step.

```
type SimulationResult s = (SimulationTable s, Seq (Time,s))
```

A.3 SRWS Monad

The executor is run inside of a reader/writer/state monad called `SRWS`. The reader's parameter is of `SolverParameters`. The state's parameter is of `SolverState`. Due to the inefficiencies of the available writer monads, a new writer monad was needed, see Section A.5. Therefore,

writer of `RWST` is set to the singular type, (), and instead the new writer is composed with the `RSWT` monad.

```
type SRWS s = RWST (SolverParameters s) () (SolverState s) (WriterT (Seq (Time,s)) Identity)

runSRWS :: SRWS s a → SolverParameters s → SolverState s
        → ((a, SolverState s, ()), Seq (Time,s))
runSRWS m r s = runIdentity $ runWriterT $ runRWST m r s
```

A.4 Executor

At the high-level, the executor takes as input a default value, a solver function, and a list of messages and gives the results. It does this by incrementally taking in each message and solving for each unique time step.

```
executor :: s → SolverFunc s → [Message] → SimulationResult s
executor def f msgs = (usedBefore ss,sc)
    where ((_,ss,_),sc) = runSRWS (simulateInc msgs) (SParams def f) emptySolverState
```

`simulateInc` examines each message individually. If it is a `MsgBegins`, an active link is added. If it is a `MsgEnds`, the corresponding active link is removed. Note that messages are order such that they are time ordered and every `MsgBegins` has a corresponding `MsgEnds` that appears later in the list. Two or more consecutive `MsgBegins` or `MsgEnds` is forbidden. Therefore, there is never the case where a link is removed that does not exist in the active links set.

Whenever it is determined that the function has seen everything of the current time step, the new time step is solved for and written. The simulator ends when there are no more remaining messages.

```
simulateInc :: [Message] → SRWS s ()
simulateInc []                          = return ()
simulateInc (MsgBegins t s d : msgs) = do
    addActiveLink s d
    when (null msgs || time (head msgs) /= t) $
        addNextTimeStep t
    simulateInc msgs
simulateInc (MsgEnds   t s d : msgs) = do
    removeActiveLink s d
    when (null msgs || time (head msgs) /= t) $
        addNextTimeStep t
    simulateInc msgs
```

To add a new time step, we execute the solver function and add the result to the current time sequence.

```
addNextTimeStep :: Time → SRWS s ()
addNextTimeStep t = do
    !s ← execSolver
    lift $ tell (t,s)
```

`execSolver` handles the memoization of the solver function. If there are no active links, the defaultValue is return. If solver has seen the same set of active links before, it returns the previous computation. Otherwise, the result is computed, saved in the simulation table, and returned.

```
execSolver :: SRWS s s
execSolver = do
    links ← gets activeLinks
    if null links then
        asks defaultValue
    else do
        br    ← haveSeenStateBefore
        case br of
            Just br' → return br'
            Nothing  → do
                f ← asks solverFunc
                let br' = f links
                modify (λ !ss → ss { usedBefore = H.insert links br' $ usedBefore ss })
                return $! br'

haveSeenStateBefore :: SRWS s (Maybe s)
haveSeenStateBefore = do
    links ← gets activeLinks
    table ← gets usedBefore
    return $! H.lookup links table
```

The necessary code to add and remove the active links from the solver state are given below.

```
addActiveLink :: TxNode → RxNode → SRWS s ()
addActiveLink s d = modify (editActiveLinks (HS.insert (txNode s,d)))

removeActiveLink :: TxNode → RxNode → SRWS s ()
removeActiveLink s d = modify (editActiveLinks (HS.delete (txNode s,d)))

editActiveLinks :: (ActiveLinks → ActiveLinks) → SolverState a → SolverState a
editActiveLinks f ss = ss { activeLinks = f $ activeLinks ss }
```

Initially, the executor's state has no active links and has not processed any previous active links.

```
emptySolverState :: SolverState a
emptySolverState = SS mempty mempty
```

A.5 Writer Monad

Due to the unavaibility of a writer monad that was strict in the state that it is writing, it is necessary to rewrite the writer monad to accomodate this. Without rewriting the writer monad, memory blows up quickly for large simulations. Note that only modifications from the original `WriterT` monad are the bind function, $\gg=$ `tell` function; and `fmap` function. The reader is referred to the *mtl* library documentation on Hackage for further information on `WriterT`.

```
newtype WriterT w m a = WriterT { unWriterT :: w → m (a, w) }

instance (Monad m) => Monad (WriterT w m) where
    m >>= f  = WriterT $ λ !w → do
        (a, !w') ← unWriterT m w
        unWriterT (f a) w'

instance (Functor m) => Functor (WriterT w m) where
    fmap f m = WriterT $ λ !w → first f <$> unWriterT m w

instance (Functor m, Monad m) => Applicative (WriterT w m) where
    pure a = WriterT $ λw → return (a, w)
    WriterT mf <*> WriterT mx = WriterT $ λw → do
        (f, w') ← mf w
        (x, w'') ← mx w'
        return (f x, w'')

runWriterT :: (Monoid w) => WriterT w m a → m (a, w)
runWriterT m = unWriterT m mempty

tell :: (Monad m) => a → WriterT (Seq a) m ()
tell a = WriterT $ λ !w' →
    let wt = w' |> a
    in wt `seq` return ((), wt)
```

Appendix B

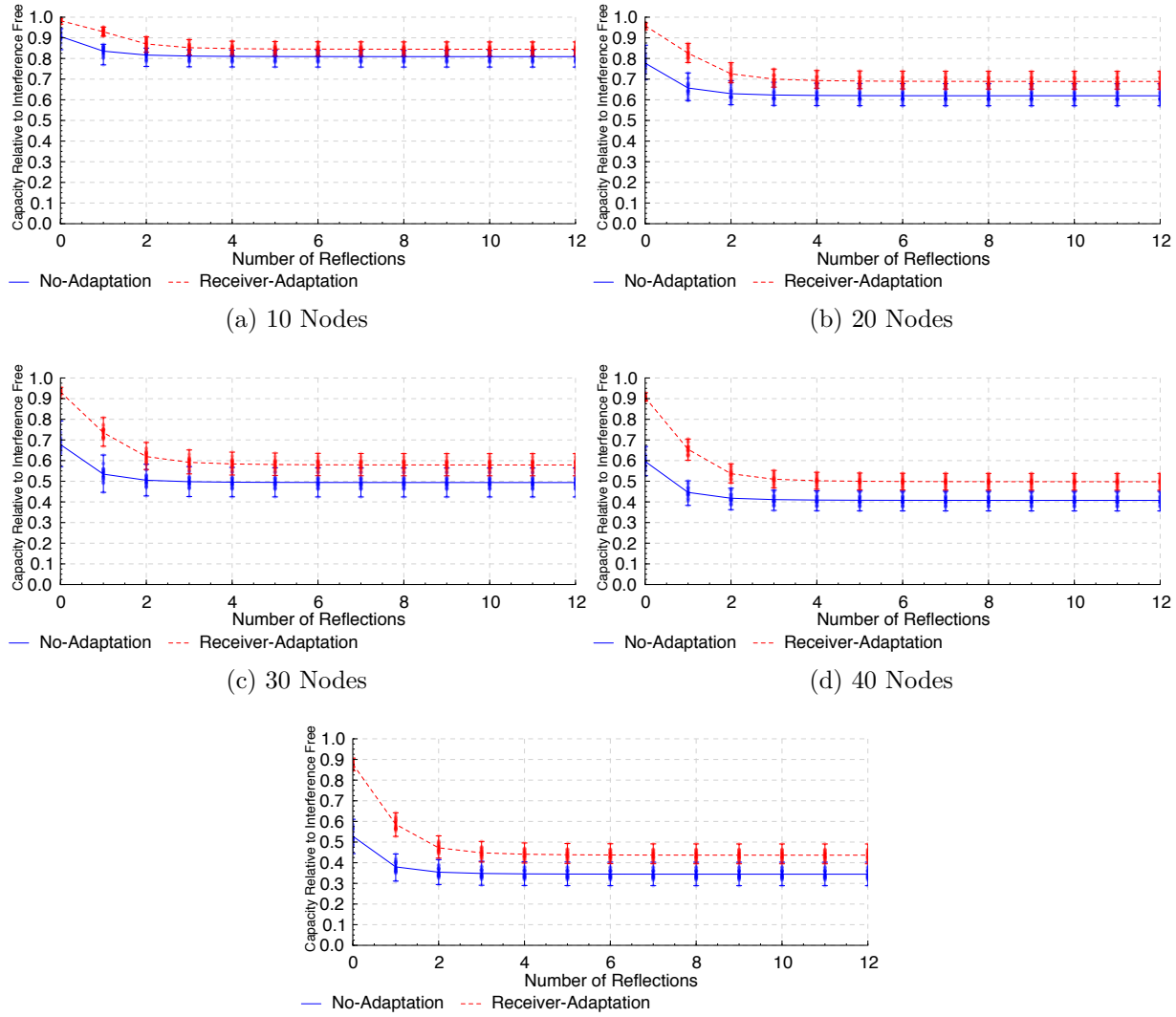
Interference Mitigation — Effect of Simulated Reflection Count

This is an appendix of Chapter 3 detailing the effect of reflections on the relative capacity. Specifically, this data was used to determine how many reflections were necessary to be simulated when running the experiments. Determining this number is rather important as ray tracing can be very computationally expensive when the number of reflections that are needed are very high.

Generally speaking, Receiver-Adaptation requires more reflections to be simulated than No-Adaptation. The reason for this is due to the fact that Receiver-Adaptation is actively trying to form nulls and therefore if too few reflections are simulated, the nulls can possibly perfectly remove interference. More reflections are necessary to make the formation of nulls non-trivial even though the reflections may be 100 dB from the line of sight path. Because the number of nulls possible is proportional to the number of antennas, one can expect that more simulated reflections are necessary to arrive at a good estimate of capacity.

Ultimately, arrays greater than 32×32 were not simulated partly due to the number of reflections that would be necessary to have an accurate estimate of capacity.

The following figures detail how the number of simulated reflections affected the result for various parameters of antenna arrays and geometries. Note that because the fixed geometry, Figure 3.5, has more than twice the maximum number nodes of the random geometries, the number of reflections necessary to simulate is also less, Figure B.6.



(a) 10 Nodes

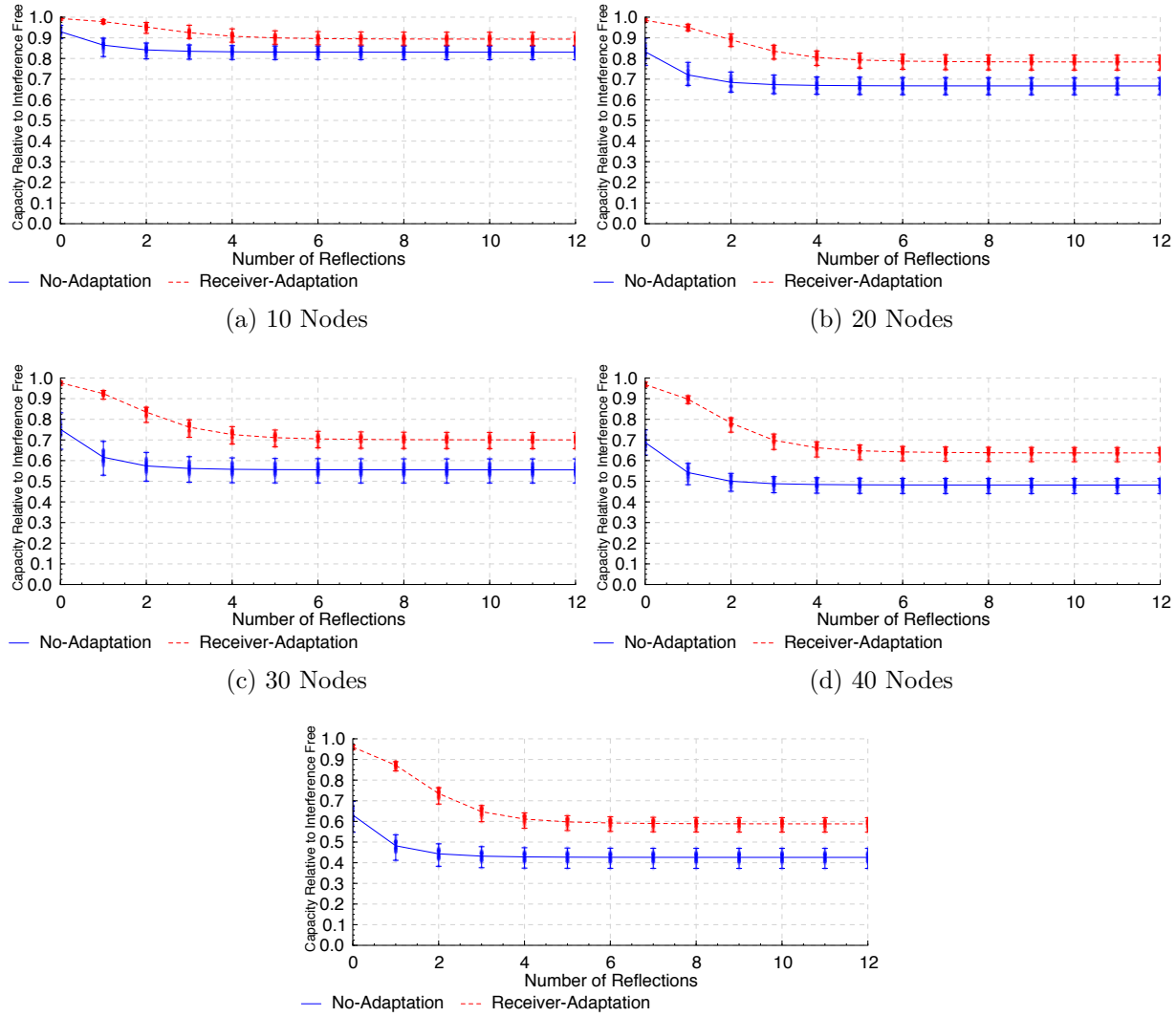
(b) 20 Nodes

(c) 30 Nodes

(d) 40 Nodes

(e) 50 Nodes

Figure B.1: 2×2 Array, Random Geometry — Effect of Simulated Reflections



(a) 10 Nodes

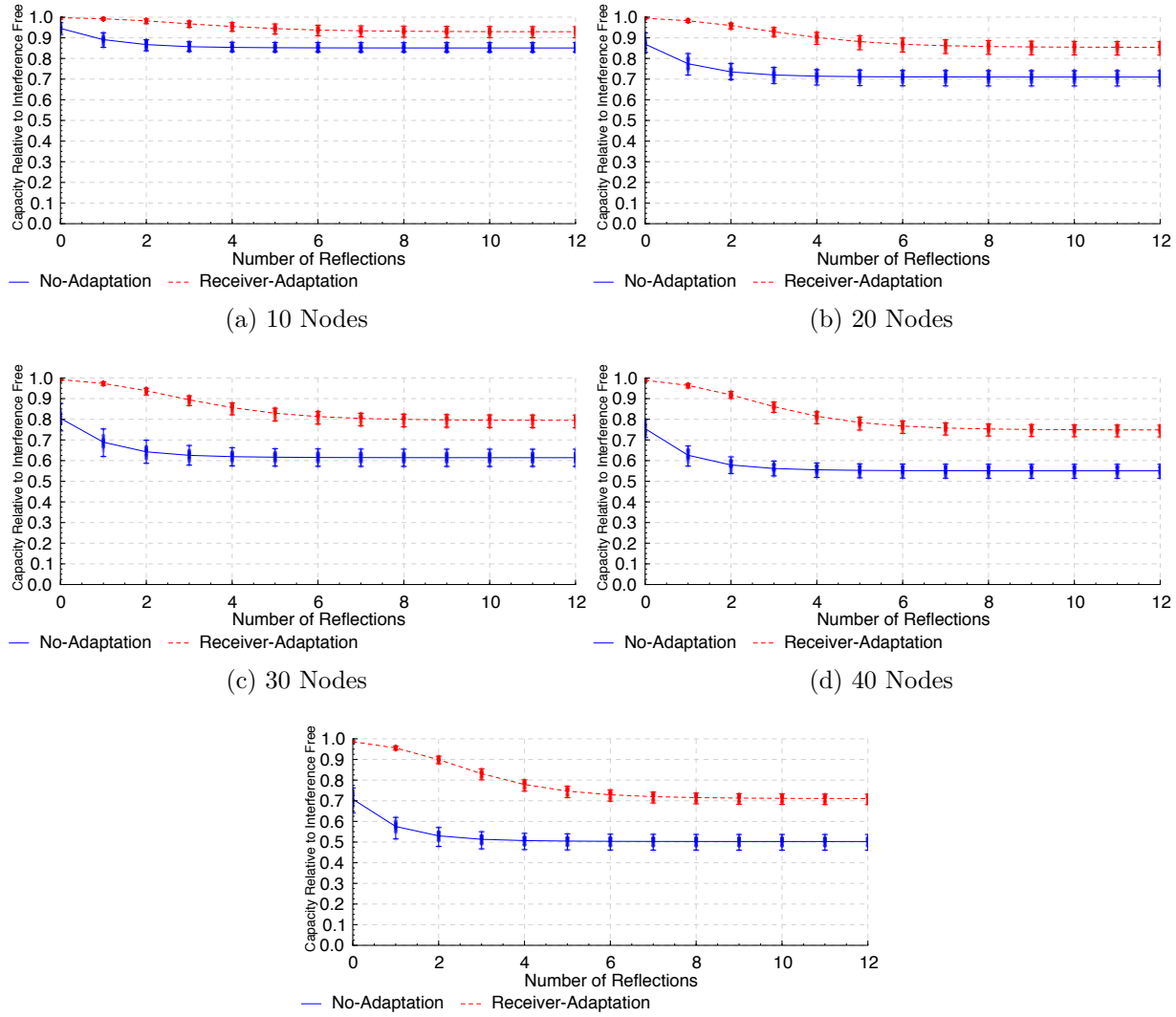
(b) 20 Nodes

(c) 30 Nodes

(d) 40 Nodes

(e) 50 Nodes

Figure B.2: 4×4 Array, Random Geometry — Effect of Simulated Reflections



(a) 10 Nodes

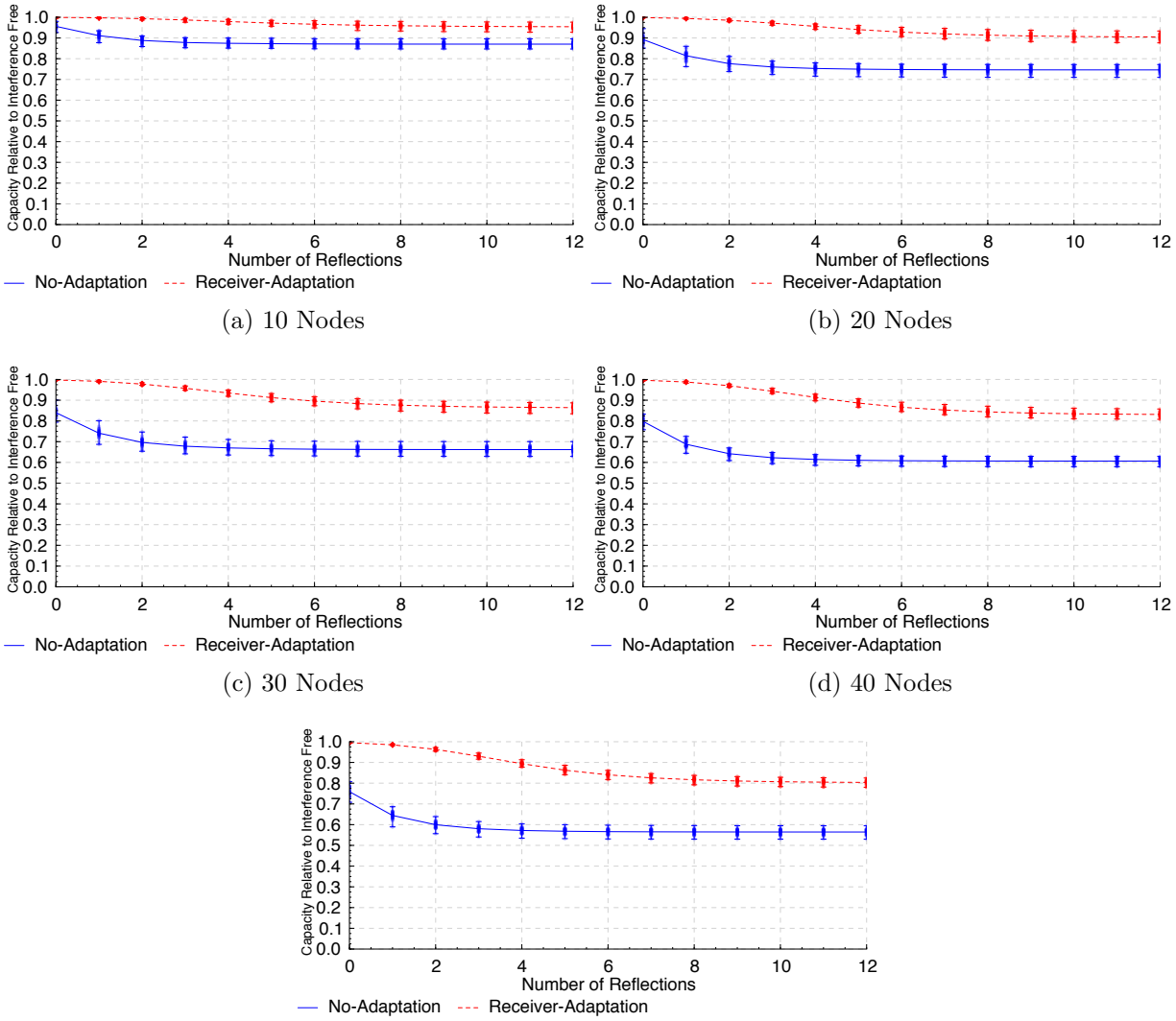
(b) 20 Nodes

(c) 30 Nodes

(d) 40 Nodes

(e) 50 Nodes

Figure B.3: 8×8 Array, Random Geometry — Effect of Simulated Reflections



(a) 10 Nodes

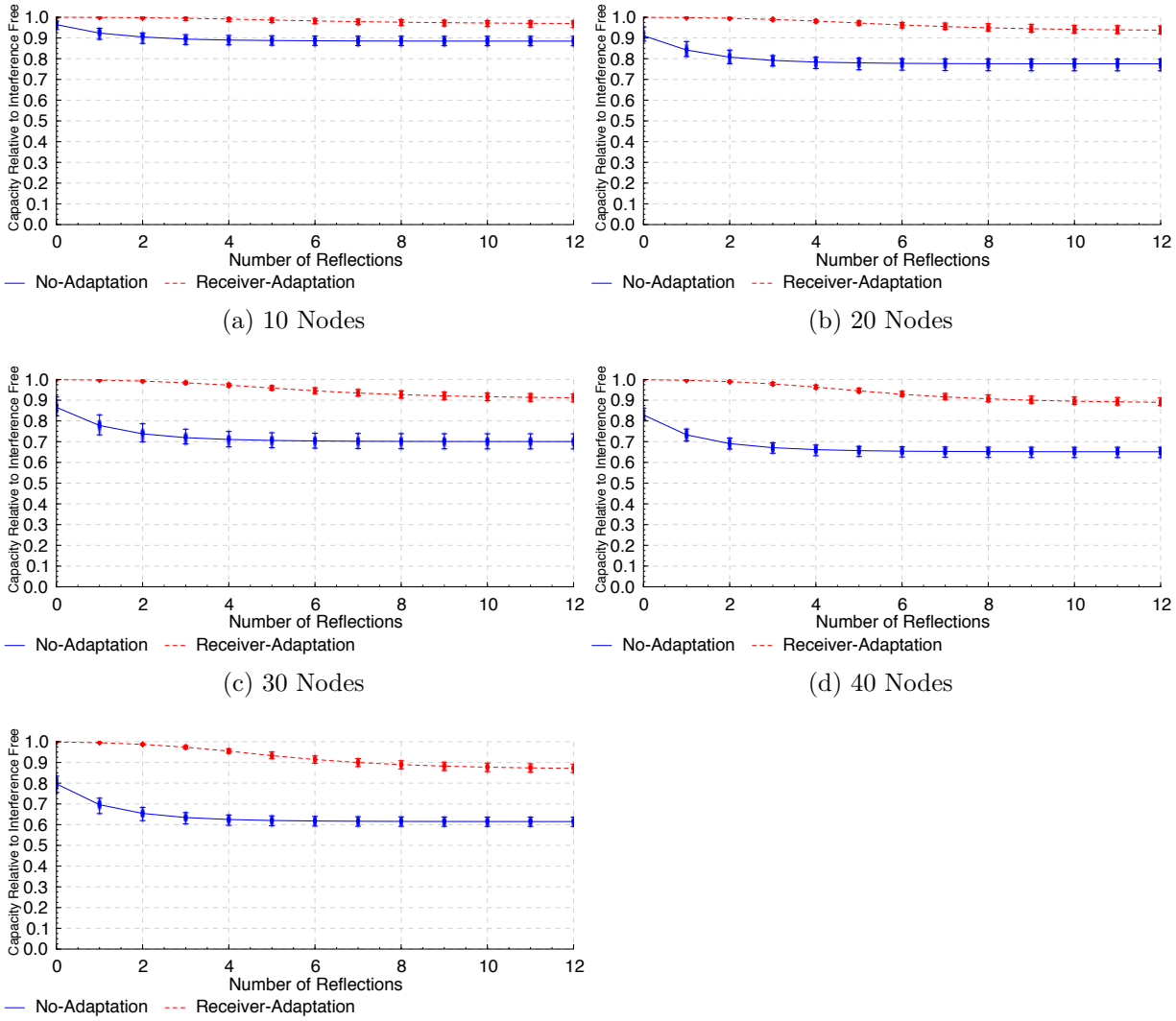
(b) 20 Nodes

(c) 30 Nodes

(d) 40 Nodes

(e) 50 Nodes

Figure B.4: 16×16 Array, Random Geometry — Effect of Simulated Reflections



(a) 10 Nodes

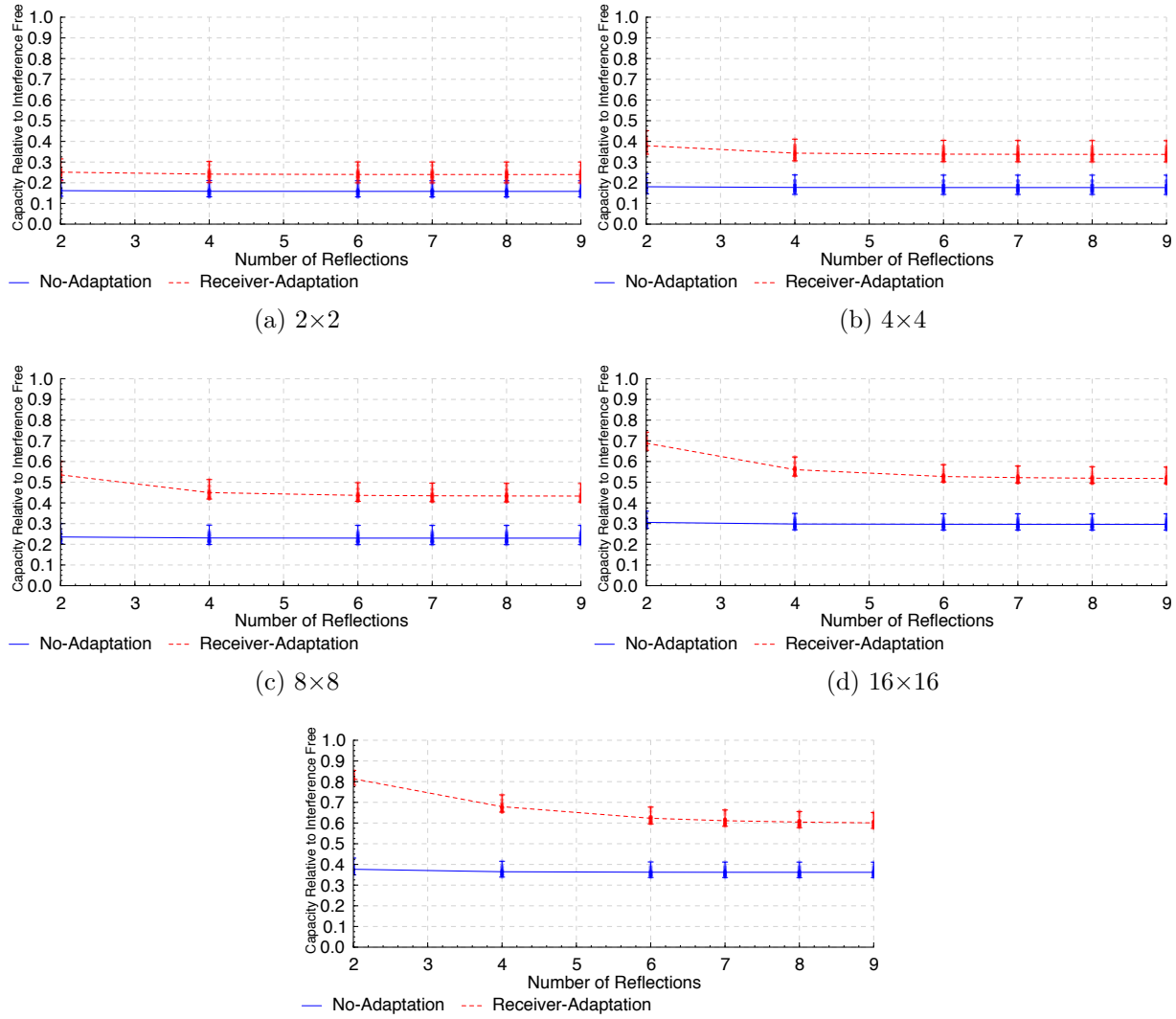
(b) 20 Nodes

(c) 30 Nodes

(d) 40 Nodes

(e) 50 Nodes

Figure B.5: 32×32 Array, Random Geometry — Effect of Simulated Reflections



(a) 2×2

(b) 4×4

(c) 8×8

(d) 16×16

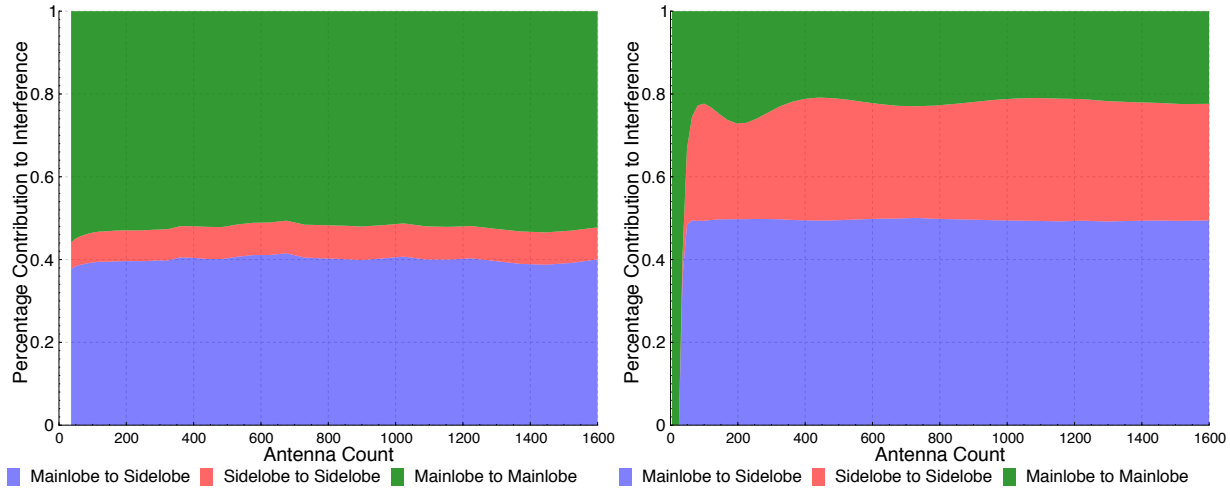
(e) 32×32

Figure B.6: Fixed Geometry — Effect of Simulated Reflections

Appendix C

Antenna Array Geometries — Sources of Expected Interference

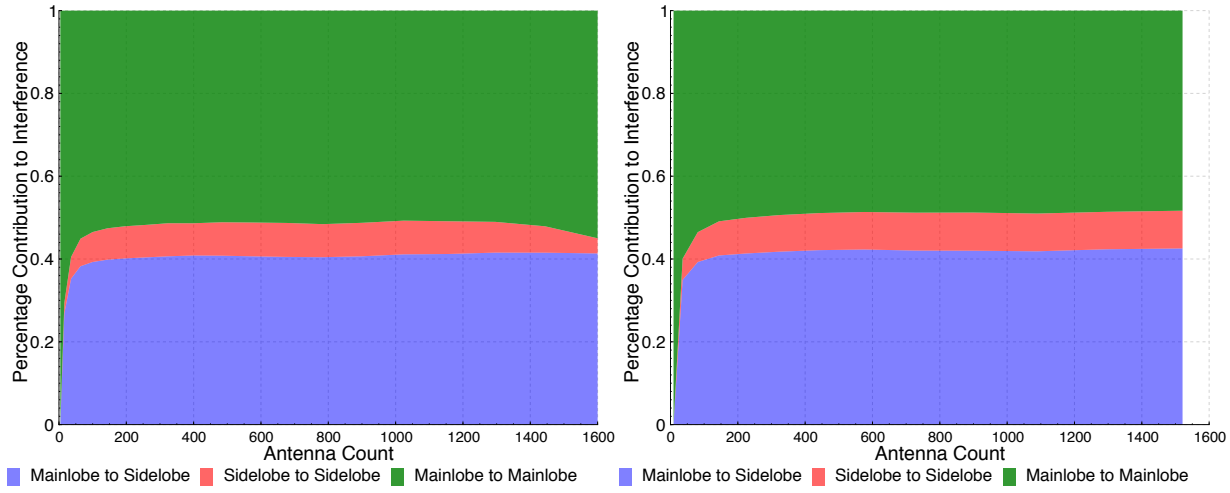
This appendix serves as additional material for the Antenna Array Geometries chapter (Chapter 4). The expected contribution of interference for a sphere is shown individually for each antenna array. These were used to determine the convergence points of each array shown in Figure 4.14.



(a) Linear Array

(b) Square Array

Figure C.1: Contribution to Interference: Linear and Square Array.



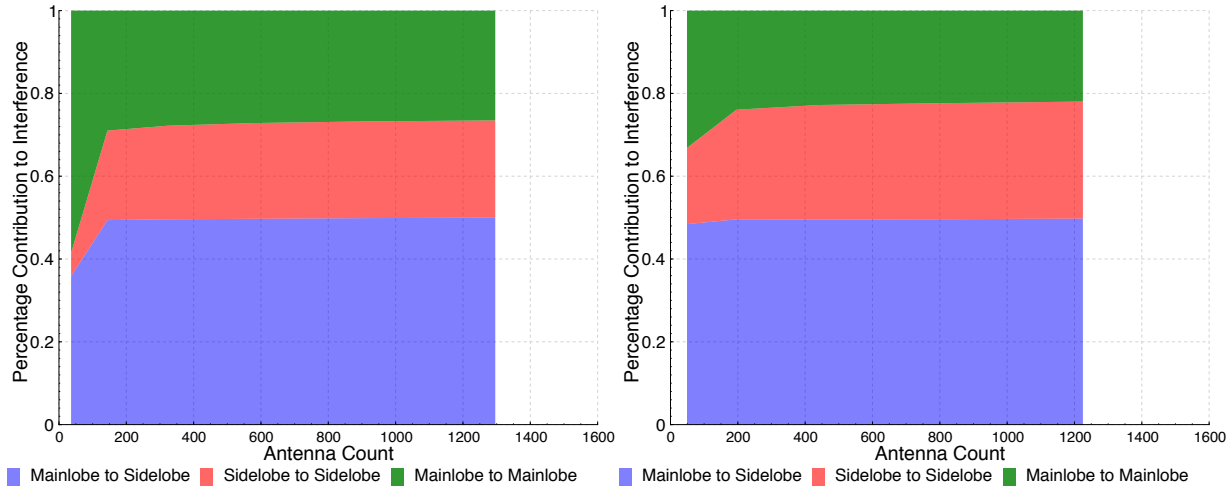
(a) Rect 2

(b) Rect 3

(c) Rect 4

(d) Rect 5

Figure C.2: Contribution to Interference: Rectangular Arrays 2x–5x



(a) Rect 6

(b) Rect 7

(c) Rect 8

(d) Rect 9

Figure C.3: Contribution to Interference: Rectangular Arrays 6x–9x

Appendix D

Listen-only Scheduling — Maximum Throughput Algorithms

This appendix gives the algorithm for calculating the theoretical maximum throughput given the assumptions in the listen-only scheduling evaluation, Section 5.5. The code is in Haskell. Max Global is a linear program that calculates the theoretical maximum throughput by solving the multi-commodity maximum flow problem with fixed paths. Max Local is a fixed point algorithm that takes into account the lack of flow control and fair arbitration at each node.

D.1 Libraries

The necessary libraries in addition to Prelude are given below. Note that `IntMap` is imported qualified as M to avoid namespace issues.

```
import Data.Ratio
import Data.IntMap(IntMap)
import qualified Data.IntMap as M
```

For Max Global, two additional libraries are used. `LinearProgram` is from a library available on Hackage by the name *glpk-hs*.

```
import Control.Monad(forM_)
import Data.LinearProgram(glpSolveVars,simplexDefaults,linCombination,execLPT
                         ,setDirection,setObjective,leqTo,varBds,LP,LinFunc,Direction(..))
```

D.2 Types

Each `Node` in the network is a unique integer value.

```
type Node = Int
```

A `Path` is a list of nodes in order from source to destination. The source node is not included in this path as it is not needed because the throughput of the receivers is being calculated not the transmitters.

```
type Path = [Node]
```

The `PathTable` is a table of all paths in the network indexed by the source node of the path.

```
type PathTable = IntMap Path
```

D.3 Max Global

Max Global is the multi-commodity maximum flow algorithm. Because the routes are fixed and not also being solved, a linear program may be used to calculate the maximum throughput of the system. The network consists of nodes, V, with P paths. Each node a capacity of 1. A path, p_i, is a boolean function indicating if the node is used on the path, $V \rightarrow 0, 1$. Every path, p_i, in the network has a demand, $d_i \in [0, 1]$. The objective function is given as maximizing the sum of demands.

$$\max \sum_{i}^{P} d_i \tag{D.1}$$

Node capacities are considered instead of edge capacities. All nodes have capacity of one. The sum of all paths through a node may not exceed the capacity of the node.

$$\forall v \in V, \sum_{i=1}^{P} p_i(v) d_i \leq 1 \tag{D.2}$$

Additionally, the demands of the paths are constrained to be $0 \leq d_i \leq 1$.

$$\forall i \in P, 0 \leq d_i \leq 1 \tag{D.3}$$

Haskell Implementation

The Haskell implementation is given below.

```
maxGlobal :: [Node] → PathTable → IO Double
maxGlobal nodes pathTable = do
    prog ← lp pathTable nodes
    return $ snd <$> glpSolveVars simplexDefaults prog

lp :: PathTable → [Node] → IO (LP String Int)
lp pathTable nodes = execLPT $ do
    setDirection Max
    setObjective $ objFun pathTable
```

```
    let equations = createEquations pathTable
    -- Constrains the demands of each node to not exceed the capacity, 1, of a node
    forM_ equations $ λeq →
        leqTo (linCombination eq) 1
    -- Constrains every node capacity to be between 0 and 1
    forM_ nodes $ λx →
        varBds (show x) 0 1

-- | Objective function
objFun :: PathTable → LinFunc String Int
objFun = linCombination . zip [1,1..] . fmap show . M.keys

-- | Create equations
createEquations :: Num a => PathTable → IntMap [(a,String)]
createEquations = M.unionsWith (++) . M.elems .
        M.mapWithKey (λk → foldr (λx → M.insertWith (++) x [(1,show k)]) M.empty)
```

D.4 Max Local

Max Local is an algorithm that takes as input a set of paths and a list of nodes and computes the maximum throughput of the network assuming every node is generating packets on every slot. The algorithm takes into account the lack of flow control and that each node will fairly allocate resources locally.

Types

In addition to the types already listed, additional types are needing for calculating Max Local. The throughput of a node or path is a rational number in the set of $[0, 1]$.

```
newtype Throughput = T { fromT :: Rational }
    deriving (Eq,Ord,Num)
```

The allocation given to a path is a rational number in the set of $[0, 1]$.

```
newtype Allocation = Alloc { fromAlloc :: Rational }
```

Path throughput, `PathThroughput`, is a tuple consisting of the amount of throughput consumed at each node on the path and the total throughput of the path.

```
type PathThroughput = ([(Node,Throughput)],Throughput)
```

The capacity left, `CapacityLeft`, represents how much of the node's throughput has not been allocated to paths yet. It is also rational number in the set of $[0, 1]$.

```
newtype CapacityLeft = CL { capacityLeft :: Rational }
```

Various tables are needed for keeping track of intermediate values. The capacity left table, `CapacityTable`, keeps track of the capacity remaining for each node in the network and is part of the fixed point function's arguments. The table is indexed by node.

```
type CapacityTable = IntMap CapacityLeft
```

The allocation table, `AllocationTable`, is the amount of allocation to each path. It is indexed by the source node of the path.

```
type AllocationTable = IntMap Allocation
```

Lastly, the throughput table, `ThroughputTable`, is the amount of throughput for each path, indexed by the source node of the path. This table is the main result for the Max Local algorithm.

```
type ThroughputTable = IntMap Throughput
```

Fixed Point Description

The result of the algorithm can be described as a fixed point on `ThroughputTable` and `CapacityTable`.

$$f :: (ThroughputTable, CapacityTable) \rightarrow (ThroughputTable, CapacityTable) \qquad \text{(D.4)}$$

On each iteration, the throughputs and the capacities left are updated. Throughputs monotonically increase, and capacities left monotonically decrease. When the input and output are equal (i.e. nothing has changed), the algorithm has reached the desired fixed point. For efficiency reasons, the algorithm is not implemented as a fixed-point, however.

Of particular note for the fixed point is that it is possible (and likely) some node's will still have non-zero capacity left. However, for every path, there will exist a node on the path that will have no capacity left.

Algorithm

The algorithm can be described in five steps.

1 For every path, give fair allocation at each node (`shareTable`)

2 Determine the throughput at every node on that path (`createThroughputs`)

3 Update the capacity of each node (`updateCapacityTable`)

4 Remove all paths that contain a node without capacity (`removedPaths`)

5 Repeat until there are no paths left.

The top level function is given below. The capacity left of each node is initialized to one, `initialCapacityTable`. The total throughput is the sum of the throughputs of each iteration.

```
maxLocal :: [Node] → PathTable → Double
maxLocal nodes = fromRational . fromT . sum . calc (initialCapacityTable nodes)

calc :: CapacityTable → PathTable → ThroughputTable
calc ct pt
    | null pt   = M.empty           -- If there are no more paths left, we are done
    | otherwise = calc' ct pt

calc' :: CapacityTable → PathTable → ThroughputTable
calc' ct pt = M.unionWith (+) (fmap snd tps) $ calc ct' $ removeMaxedPaths ct' pt
    where tps = createThroughputs (shareTable pt ct) pt
          ct' = updateCapacityTable tps ct

initialCapacityTable :: [Node] → CapacityTable
initialCapacityTable = M.fromList . fmap (λx → (x,CL 1))
```

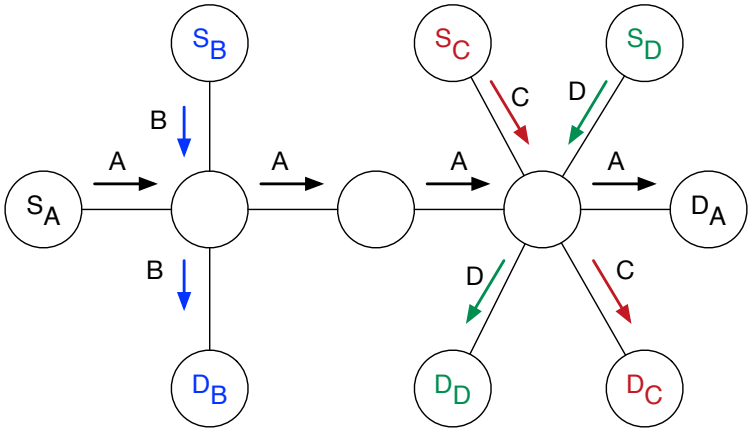
The first step operates under the assumption that every path that enters the node will use as much as capacity as possible, and therefore the fair way to distribute the allocations is to give each node an equal amount. In Figure D.1, there is an example of allocations assuming that each node has a capacity of one left. Each node in the path is given an equal share at intermediate and destination nodes. In this example, the node that shares paths A and B gives each path 1/2. The node that shares routes A, B, and C gives out 1/3.

The way the allocation is implemented is that for every node in every path add one to it. Assign the allocation to be the remaining capacity multiplied by the inverse of the total at each node.

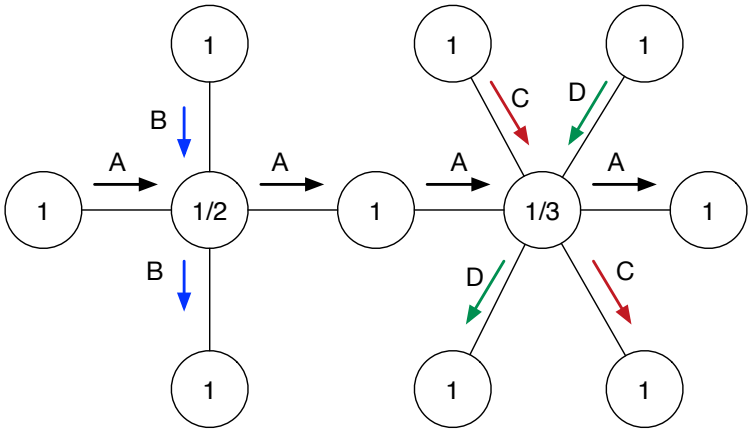
```
shareTable :: PathTable → CapacityTable → AllocationTable
shareTable pathTable ct = setCapacity $ M.fromListWith (+)
                                      $ zip (concat $ M.elems pathTable) [1,1..]
    where setCapacity = M.mapWithKey (λk x → Alloc $ capacityLeft (ct M.! k) * (1 % x))
```

The second step is to determine the actual throughput that the path will actually use keeping in mind that there is no flow control. The throughput of the entire path is the minimum of the allocations. However, what the path will use at a given node will be based on how much has been removed previously. Figure D.2 shows an example of determining the throughputs. Initially, the path has allocations of 1/3, 1/4, 1/2, 1/5, and 1. The throughput of the path is the minimum, 1/5. However, before it reaches this bottleneck, it will consume as much as the throughput of the previous bottleneck. Therefore, the used capacity is 1/3, 1/4, 1/4, 1/5, and 1/5. Notice that throughputs can only go down because of bottlenecks.

The implementation propagates the throughputs based on the allocations determined in the previous step. The propagation is done in order from source node to destination node keeping track of the current throughput. If the allocation is smaller than the throughput, the throughput is updated. Otherwise, the current throughput is kept.



(a) Set of Paths



(b) Node Allocations

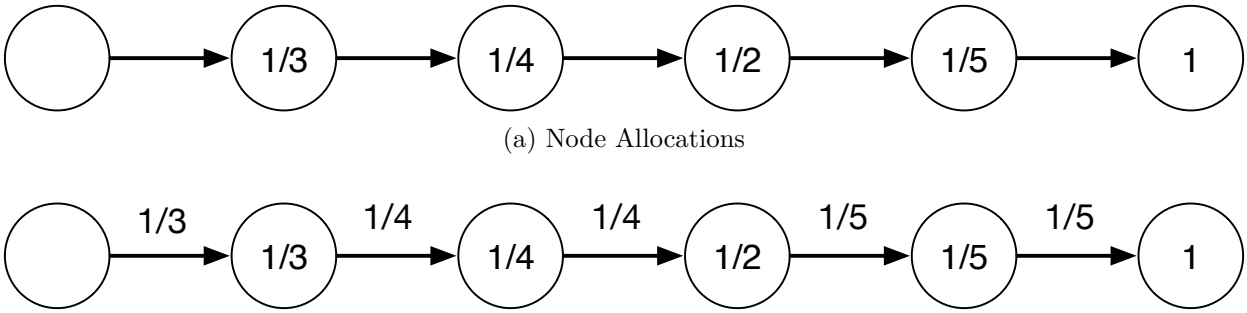
Figure D.1: Node Allocation from Paths Example

```
createThroughputs :: AllocationTable → PathTable → IntMap PathThroughput
createThroughputs allocs = fmap (pathThroughput allocs)

pathThroughput :: AllocationTable → Path → PathThroughput
pathThroughput allocs = setCapacities . lookupAllocs
    where lookupAllocs = fmap (λx → (x, T $ fromAlloc $ allocs M.! x))
          setCapacities = foldl propagateThroughput ([],1)
          propagateThroughput (xs,c) o@(n,a)
            | c > a     = (o:xs,a)
            | otherwise = ((n,c):xs,c)
```

Based on the throughputs calculated, the new capacities left is updated for each node by subtracting the new used throughput from the capacity.

```
updateCapacityTable :: IntMap PathThroughput → CapacityTable → CapacityTable
```



(a) Node Allocations

(b) Throughputs Specified

Figure D.2: Determining Path Throughputs Example

```
updateCapacityTable tps ct = M.unionWith merge ct $ fmap (CL . sum) $ convert $ M.elems tps
    where convert = M.fromListWith (++) . concatMap (fmap (λ(n,r) → (n,[fromT r])) . fst)
          merge (CL x) (CL y) = CL (x − y)
```

Lastly, any paths that contain a node without any capacity left are removed from the path table.

```
removeMaxedPaths :: CapacityTable → PathTable → PathTable
removeMaxedPaths ct = M.filter (not . any ((==0) . capacityLeft . (ct M.!)))
```

The algorithm is guaranteed to terminate because at every iteration at least one node will have no capacity left. This is due to the fact that node capacities are initialized to one, and there must be at least one bottleneck (at least one minimum) when propagating the throughputs. Therefore, at least one path will be removed from the table on every iteration. Given a finite set of paths of a finite number of nodes each, the algorithm will terminate in finite time.

The algorithm implements the maximum throughput because on every iteration any path that can use the capacity of the node will. It takes into account local fairness as each node gives equal capacity to the remaining paths.

Bibliography

[1] Abbas Abbaspour-Tamijani and Kamal Sarabandi. "An affordable millimeter-wave beam-steerable antenna using interleaved planar subarrays". In: *IEEE Transactions on Antennas and Propagation* 51.9 (2003), pp. 2193–2202.

[2] S. Alalusi and R. Brodersen. "A 60GHz Phased Array in CMOS". In: *IEEE Custom Integrated Circuits Conference 2006*. Sept. 2006, pp. 393–396. DOI: 10.1109/CICC.2006.320957.

[3] Arthur Appel. "Some Techniques for Shading Machine Renderings of Solids". In: *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*. AFIPS '68 (Spring). Atlantic City, New Jersey: ACM, 1968, pp. 37–45. DOI: 10.1145/1468075.1468082. URL: http://doi.acm.org/10.1145/1468075.1468082.

[4] A. Arvanitis et al. "Capacity Study of a Multiple Element Antenna Configuration in an Indoor Wireless Channel at 60 GHz". In: *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*. Apr. 2007, pp. 609–613. DOI: 10.1109/VETECS.2007.136.

[5] Roger Baig and et.al. "Guifi.Net, a Crowdsourced Network Infrastructure Held in Common". In: *Comput. Netw.* 90.C (Oct. 2015), pp. 150–165. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2015.07.009. URL: https://doi.org/10.1016/j.comnet.2015.07.009.

[6] T. Baykas et al. "IEEE 802.15.3c: the first IEEE wireless standard for data rates over 1 Gb/s". In: *Communications Magazine, IEEE* 49.7 (July 2011), pp. 114–121. ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5936164.

[7] O. Bazan and M. Jaseemuddin. "A Survey On MAC Protocols for Wireless Adhoc Networks with Beamforming Antennas". In: *Communications Surveys Tutorials, IEEE* 14.2 (Feb. 2012), pp. 216–239. ISSN: 1553-877X. DOI: 10.1109/SURV.2011.041311.00099.

[8] Peter Joseph Bevelacqua. "Antenna Arrays: Performance Limits And Geometry". PhD thesis. Arizona state university, 2008.

[9] Azzedine Boukerche et al. "Routing protocols in ad hoc networks: A survey". In: *Computer networks* 55.13 (2011), pp. 3032–3080.

[10] Government of Canada. *Spectrum Utilization Policy for Licence Exempt Wireless Devices in the Bands 46.7-46.9 GHz, 57-64 GHz and 76-77 GHz, Industrie Canada, doc SP-47 GHz*. 2001.

[11] Dah-Ming Chiu and Raj Jain. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks". In: *Comput. Netw. ISDN Syst.* 17.1 (June 1989), pp. 1–14. ISSN: 0169-7552. DOI: 10.1016/0169-7552(89)90019-6. URL: http://dx.doi.org/10.1016/0169-7552(89)90019-6.

[12] Romit Roy Choudhury and Nitin H. Vaidya. "Impact of Directional Antennas on Ad Hoc Routing". In: *Personal Wireless Communications*. Ed. by Marco Conti et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 590–600. ISBN: 978-3-540-39867-7.

[13] L. Cimini. "Analysis and Simulation of a Digital Mobile Channel Using Orthogonal Frequency Division Multiplexing". In: *IEEE Transactions on Communications* 33.7 (July 1985), pp. 665–675. ISSN: 0090-6778. DOI: 10.1109/TCOM.1985.1096357.

[14] Peter Clarke. "Lattice 60GHz modules allow beam steering". In: (Dec. 2017). URL: http://www.eenewsanalog.com/news/lattice-60ghz-modules-allow-beam-steering.

[15] Alan Colvin. "CSMA with collision avoidance". In: *Computer Communications* 6.5 (1983), pp. 227–235.

[16] Federal Communications Commission. *Revision of Part 15 of the Commission's Rules Regarding Operation in the 57-64 GHz Band*. 2013.

[17] C. H. Doan et al. "Design considerations for 60 GHz CMOS radios". In: *IEEE Communications Magazine* 42.12 (Dec. 2004), pp. 132–140. ISSN: 0163-6804. DOI: 10.1109/MCOM.2004.1367565.

[18] ECMA. *Standard ECMA-387 High Rate 60GHz PHY, MAC, and PALs*. Tech. rep. Dec. 2010. URL: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-387.pdf.

[19] Eero. *https://eero.com*. 2018.

[20] Office of Engineering and Technology - Federal Communications Commission. *Millimeter Wave Propagation: Spectrum Management Implications*. July 1997.

[21] Electronic Communications Committee (ECC) within the European Conference of Postal and Telecommunications Administrations (CEPT). *Use of the 57–64 GHz Frequency Band for Point-to-Point Fixed Wireless Systems*. 2009.

[22] Shoichiro Fukao et al. "The MU radar with an active phased array system: 2. In-house equipment". In: *Radio Science* 20.6 (1985), pp. 1169–1176. DOI: 10.1029/RS020i006p01169. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/RS020i006p01169. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/RS020i006p01169.

[23] J. M. Gilbert et al. "A 4-Gbps Uncompressed Wireless HD A/V Transceiver Chipset". In: *IEEE Micro* 28.2 (Mar. 2008), pp. 56–64. ISSN: 0272-1732. DOI: 10.1109/MM.2008.20.

[24] S. G. Glisic, R. Rao, and L. B. Milstein. "The effect of imperfect carrier sensing on nonpersistent carrier sense multiple access". In: *IEEE International Conference on Communications, Including Supercomm Technical Sessions*. Apr. 1990, 1266–1269 vol.3. DOI: 10.1109/ICC.1990.117274.

[25] Eric Groft et al. *Smart meter parking system*. US Patent App. 11/179,779. Jan. 2007.

[26] P. Gupta and P. R. Kumar. "The capacity of wireless networks". In: *IEEE Transactions on Information Theory* 46.2 (Mar. 2000), pp. 388–404. ISSN: 0018-9448. DOI: 10.1109/18.825799.

[27] Z. J. Haas. "A new routing protocol for the reconfigurable wireless networks". In: *Proceedings of ICUPC 97 - 6th International Conference on Universal Personal Communications*. Vol. 2. Oct. 1997, 562–566 vol.2. DOI: 10.1109/ICUPC.1997.627227.

[28] Jacques Hadamard. "Resolution d'une question relative aux determinants". In: *Bull. des sciences math.* 2 (1893), pp. 240–246.

[29] B. Huder and W. Menzel. "Flat printed reflector antenna for mm-wave applications". In: *Electronics Letters* 24.6 (Mar. 1988), pp. 318–319. ISSN: 0013-5194. DOI: 10.1049/el:19880214.

[30] "IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band". In: *IEEE Std 802.11ad-2012 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012 and IEEE Std 802.11aa-2012)* (Dec. 2012), pp. 1–628. DOI: 10.1109/IEEESTD.2012.6392842.

[31] "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture". In: *IEEE Std 802-2001 (Revision of IEEE Std 802-1990)* (Feb. 2002), pp. 1–48. DOI: 10.1109/IEEESTD.2002.93395.

[32] European Telecommunications Standards Institute. *Broadband Radio Access Networks (BRAN); 60 GHz Multiple-Gigabit WAS/RLAN Systems;Harmonized EN covering the essential requirements of article 3.2 of the RTTE Directive*. 2011.

[33] P. Jacquet et al. "Optimized link state routing protocol for ad hoc networks". In: *Proceedings. IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century.* Dec. 2001, pp. 62–68. DOI: 10.1109/INMIC.2001.995315.

[34] G. Jakllari, W. Luo, and S.V. Krishnamurthy. "An Integrated Neighbor Discovery and MAC Protocol for Ad Hoc Networks Using Directional Antennas". In: *Wireless Communications, IEEE Transactions on* 6.3 (Mar. 2007), pp. 1114–1024. ISSN: 1536-1276. DOI: 10.1109/TWC.2007.05471.

[35] David B. Johnson and David A. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks". In: *Mobile Computing*. Ed. by Tomasz Imielinski and Henry F. Korth. Boston, MA: Springer US, 1996, pp. 153–181. ISBN: 978-0-585-29603-6. DOI: 10.1007/978-0-585-29603-6_5. URL: https://doi.org/10.1007/978-0-585-29603-6_5.

[36] Phil Karn et al. "MACA-a new channel access method for packet radio". In: *ARRL/CRRL Amateur radio 9th computer networking conference*. Vol. 140. London, Canada. 1990, pp. 134–140.

[37] Brad Karp and H. T. Kung. "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks". In: *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*. MobiCom '00. Boston, Massachusetts, USA: ACM, 2000, pp. 243–254. ISBN: 1-58113-197-6. DOI: 10.1145/345910.345953. URL: http://doi.acm.org/10.1145/345910.345953.

[38] P Kaufmann et al. "The use of the large mm-wave antenna at Itapetinga in high-sensitivity solar research". In: *Solar Physics* 78.2 (1982), pp. 389–399.

[39] L. Kleinrock and F. Tobagi. "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics". In: *IEEE Transactions on Communications* 23.12 (Dec. 1975), pp. 1400–1416. ISSN: 0090-6778. DOI: 10.1109/TCOM.1975.1092768.

[40] Leonard Kleinrock. *Communication nets: Stochastic message flow and delay*. Courier Corporation, 1964.

[41] Young-Bae Ko and Nitin H. Vaidya. "Location-Aided Routing (LAR) in mobile ad hoc networks". In: *Wireless Networks* 6.4 (Sept. 2000), pp. 307–321. ISSN: 1572-8196. DOI: 10.1023/A:1019106118419. URL: https://doi.org/10.1023/A:1019106118419.

[42] S.S. Kulkarni and C. Rosenberg. "DBSMA: a MAC protocol for multi-hop ad-hoc networks with directional antennas". In: *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*. Vol. 2. Sept. 2005, 1371–1377 Vol. 2. DOI: 10.1109/PIMRC.2005.1651664.

[43] C.-P. Lim, M. Lee, and R. J. et.al. Burkholder. "60 GHz Indoor Propagation Studies for Wireless Communications Based on a Ray-tracing Method". In: *EURASIP J. Wirel. Commun. Netw.* 2007.1 (Jan. 2007), pp. 49–49. ISSN: 1687-1472. DOI: 10.1155/2007/73928.

[44] A. Maltsev and et. al. "Experimental investigations of 60 GHz WLAN systems in office environment". In: *IEEE Journal on Selected Areas in Communications* 27.8 (Oct. 2009), pp. 1488–1499. ISSN: 0733-8716. DOI: 10.1109/JSAC.2009.091018.

[45] T. Manabe, Y. Miura, and T. Ihara. "Effects of antenna directivity and polarization on indoor multipath propagation characteristics at 60 GHz". In: *IEEE Journal on Selected Areas in Communications* 14.3 (Apr. 1996), pp. 441–448. ISSN: 0733-8716. DOI: 10.1109/49.490229.

[46] R. Mudumbai, S. Singh, and U. Madhow. "Medium Access Control for 60 GHz Outdoor Mesh Networks with Highly Directional Links". In: *INFOCOM 2009, IEEE*. Apr. 2009, pp. 2871–2875. DOI: 10.1109/INFCOM.2009.5062249.

[47] Sophocles J. Orfanidis. *Electromagnetic Waves and Antennas*. 2016.

[48] C. E. Perkins and E. M. Royer. "Ad-hoc on-demand distance vector routing". In: *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*. Feb. 1999, pp. 90–100. DOI: 10.1109/MCSA.1999.749281.

[49] Charles E. Perkins and Pravin Bhagwat. "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers". In: *SIGCOMM Comput. Commun. Rev.* 24.4 (Oct. 1994), pp. 234–244. ISSN: 0146-4833. DOI: 10.1145/190809.190336. URL: http://doi.acm.org/10.1145/190809.190336.

[50] M. Peter, W. Keusgen, and R. Felbecker. "Measurement and Ray-Tracing Simulation of the 60 GHz Indoor Broadband Channel: Model Accuracy and Parameterization". In: *The Second European Conference on Antennas and Propagation, EuCAP 2007*. Nov. 2007, pp. 1–8. DOI: 10.1049/ic.2007.1555.

[51] Olinde Rodrigues. "Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace: et de la variation des cordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire". In: *J. Math. Pures Appl.* (1840).

[52] Claude Elwood Shannon. "A mathematical theory of communication". In: *Bell system technical journal* 27.3 (July 1948), pp. 379–423. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[53] B. A. Sharp, E. A. Grindrod, and D. A. Camm. "Hybrid TDMA/CSMA protocol for self managing packet radio networks". In: *Proceedings of ICUPC '95 - 4th IEEE International Conference on Universal Personal Communications*. Nov. 1995, pp. 929–933. DOI: 10.1109/ICUPC.1995.497145.

[54] A. K. Singh et al. "Hierarchical routing protocols in WSN: A brief survey". In: *2017 3rd International Conference on Advances in Computing,Communication Automation (ICACCA) (Fall)*. Sept. 2017, pp. 1–6. DOI: 10.1109/ICACCAF.2017.8344659.

[55] S. Singh, R. Mudumbai, and U. Madhow. "Distributed Coordination with Deaf Neighbors: Efficient Medium Access for 60 GHz Mesh Networks". In: *INFOCOM, 2010 Proceedings IEEE*. Mar. 2010, pp. 1–9. DOI: 10.1109/INFCOM.2010.5461996.

[56] A. Spyropoulos and C. S. Raghavendra. "Capacity bounds for ad-hoc networks using directional antennas". In: *Communications, 2003. ICC '03. IEEE International Conference on*. Vol. 1. May 2003, 348–352 vol.1. DOI: 10.1109/ICC.2003.1204197.

[57] Sanjib Sur et al. "60 GHz Indoor Networking Through Flexible Beams: A Link-Level Profiling". In: *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '15. Portland, Oregon, USA: ACM, 2015, pp. 71–84. ISBN: 978-1-4503-3486-0. DOI: 10.1145/2745844.2745858. URL: http://doi.acm.org/10.1145/2745844.2745858.

[58] James Joseph Sylvester. "LX. Thoughts on inverse orthogonal matrices, simultaneous signsuccessions, and tessellated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work, and the theory of numbers". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 34.232 (1867), pp. 461–475.

[59] Wojciech Tadej and Karol Życzkowski. "A concise guide to complex Hadamard matrices". In: *Open Systems & Information Dynamics* 13.2 (2006), pp. 133–177.

[60] Y. Takatsuka et al. "A MAC protocol for directional hidden terminal and minor lobe problems". In: *Wireless Telecommunications Symposium, 2008. WTS 2008*. Apr. 2008, pp. 210–219. DOI: 10.1109/WTS.2008.4547567.

[61] O. Tipmongkolsilp, S. Zaghloul, and A. Jukan. "The Evolution of Cellular Backhaul Technologies: Current Issues and Future Trends". In: *IEEE Communications Surveys Tutorials* 13.1 (Jan. 2011), pp. 97–113. ISSN: 1553-877X. DOI: 10.1109/SURV.2011.040610.00039.

[62] Fouad Tobagi and Leonard Kleinrock. "Packet switching in radio channels: part II–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution". In: *IEEE Transactions on communications* 23.12 (1975), pp. 1417–1433.

[63] Rob Van Gerwen, Saskia Jaarsma, and Rob Wilhite. "Smart metering". In: *Leonardo-energy. org* 9 (2006).

[64] Andrew J Viterbi. *CDMA: principles of spread spectrum communication*. Addison Wesley Longman Publishing Co., Inc., 1995.

[65] Jingjing Wang et al. "Capacity of 60 GHz Wireless Communication Systems over Fading Channels." In: *JNW* 7.1 (2012), pp. 203–209.

[66] Bernard Widrow and Samuel D. Stearns. *Adaptive Signal Processing*. Pearson Education, 1985. ISBN: 9788131705322.

[67] WiGig. *http://www.wi-fi.org/discover-wi-fi/wigig-certified*. 2015.

[68] Virginia Vassilevska Williams. "Multiplying Matrices Faster than Coppersmith-Winograd". In: *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*. STOC '12. New York, New York, USA: ACM, 2012, pp. 887–898. ISBN: 978-1-4503-1245-5. DOI: 10.1145/2213977.2214056.

[69] "Wilocity Bringing WiGig To Your Desk, Lap, Home and Office". In: (Jan. 2013). URL: https://www.anandtech.com/show/6646/wilocity-bringing-wigig-to-your-desk-lap-home-and-office.

[70] WirelessHD. *WirelessHD Specification Version 1.1 Overview*. Tech. rep. May 2010. URL: http://www.wirelesshd.org/pdfs/WirelessHD-Specification-Overview-v1.1May2010.pdf.

[71] Su Yi, Yong Pei, and Shivkumar Kalyanaraman. "On the Capacity Improvement of Ad Hoc Wireless Networks Using Directional Antennas". In: *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*. MobiHoc '03. Annapolis, Maryland, USA: ACM, 2003, pp. 108–116. ISBN: 1-58113-684-6. DOI: 10.1145/778415.778429.

[72] S. Zihir et al. "60-GHz 64- and 256-Elements Wafer-Scale Phased-Array Transmitters Using Full-Reticle and Subreticle Stitching Techniques". In: *IEEE Transactions on Microwave Theory and Techniques* 64.12 (Dec. 2016), pp. 4701–4719. ISSN: 0018-9480. DOI: 10.1109/TMTT.2016.2623948.