Practical Null Steering in Millimeter Wave Networks

Sohrab Madani*, Suraj Jog*, Jesus O. Lacruz†, Joerg Widmer†, Haitham Hassanieh*
*University of Illinois at Urbana Champaign, †IMDEA Networks

Abstract – Millimeter wave (mmWave) is playing a central role in pushing the performance and scalability of wireless networks by offering huge bandwidth and extremely high data rates. Millimeter wave radios use phased array technology to modify the antenna beam pattern and focus their power towards the transmitter or receiver. In this paper, we explore the practicality of modifying the beam pattern to suppress interference by creating nulls, i.e. directions in the beam pattern where almost no power is received. Creating nulls in practice, however, is challenging due to the fact that practical mmWave phased arrays offer very limited control in setting the parameters of the beam pattern and suffer from hardware imperfections which prevent us from nulling interference.

We introduce Nulli-Fi, the first practical mmWave null steering system. Nulli-Fi combines a novel theoretically optimal algorithm that accounts for limitations in practical phased arrays with a discrete optimization framework that overcomes hardware imperfections. Nulli-Fi also introduces a fast null steering protocol to quickly null new unforeseen interferers. We implement and extensively evaluate Nulli-Fi using commercial off-the-shelf 60 GHz mmWave radios with 16-element phased arrays transmitting IEEE 802.11ad packets [33] . Our results show that Nulli-Fi can create nulls that reduce interference by up to 18 dB even when the phased array offers only 4 bits of control. In a network with 10 links (20 nodes), Nulli-Fi's ability to null interference enables 2.68× higher total network throughput compared to recent past work.

1 Introduction

Millimeter wave (mmWave) networks introduced a major leap in data rates and scalability for 5G cellular networks, next generation wireless LANs, and IoT devices [10, 41, 46]. At the heart of millimeter wave technology are phased arrays which can focus the power of the antenna beam pattern in real-time towards the client to compensate for the large attenuation of mmWave signals. At mmWave frequencies ($\geq$ 24 GHz), phased arrays can fit many antennas into a small area due to the mm-scale wavelength of the signal [63], enabling very narrow directional beams as shown in Fig. 1. Ideally, using narrow beams would shield a mmWave device from interference outside the main direction (main lobe) of its beam. However, phased arrays suffer from side-lobe leakage as shown in Fig. 1(a). Hence, they still receive interfering signals even if these signals come from directions outside the main lobe. Past work has shown that side-lobes can lead to a significant amount of interference which can degrade the data rate and in dense networks reduce the total network throughput to half (by up to 18 Gbps) [14, 27, 44, 55, 61].

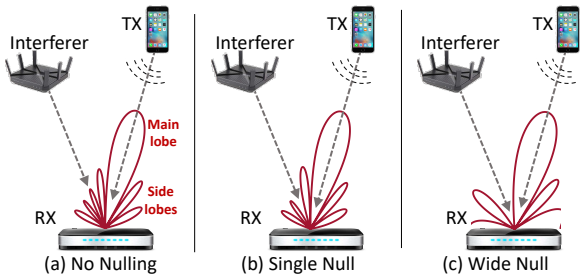


Figure 1: Directional beams in mmWave networks

To address the above problem, we leverage the fact that phased array beam patterns exhibit nulls, directions in the beam pattern where the transmitted or received power is suppressed as shown in Fig. 2(b). Thus, we can substantially reduce interference by having a null in the direction of the interferer. However, simply shifting the beam pattern to align the null with the interferer can misalign the main lobe and lead to worse performance as we show in section 6. Hence, we must create a new beam pattern to introduce a null in the direction of the interferer while preserving the alignment of the main lobe as shown in Fig. 1(b). This problem is commonly referred to as null steering.

Past mmWave systems research has mainly focused on beam alignment and steering [13, 17, 20, 40, 51, 56, 65], i.e. creating and steering the main lobe of the beam. Creating and steering nulls, however, while ensuring the main lobe is preserved is significantly harder. To better understand why, consider the phased array diagram shown in Fig. 2(a). The beam pattern of the array is created by modifying complex weights applied to each antenna element of the phased array. These complex weights alter the magnitude and phase of the signal received on each antenna. We can adjust these weights to align signals on the antennas coming from a certain direction to sum constructively creating a main lobe as shown in Fig. 2(d). In contrast, to create a null, we must ensure that the signals sum up destructively to cancel each other.

Setting the complex weights to ensure the signals from the direction of the interferer cancel each other while signals from the main lobe direction continue to sum up constructively is challenging in practice for several reasons. First, commercial mmWave phased arrays only allow us to change the phase of the complex weight but do not offer any control over the amplitude [11, 64]. While it is sufficient to rotate the phase of the signals to ensure they sum up constructively, it is hard to ensure signals cancel each other without modifying their

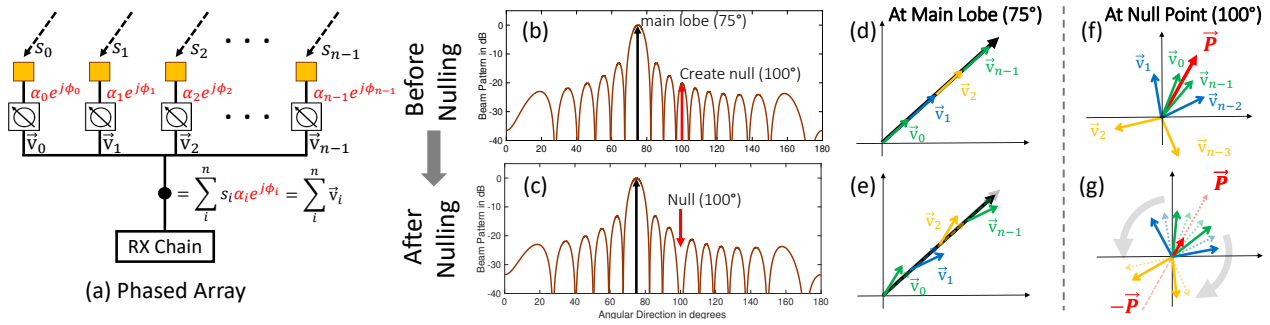


Figure 2: (a) phased arrays weight combination, (b-g) Nulli-Fi's Nulling Algorithm

amplitude. This is further complicated by the fact that phase control in practical arrays is highly quantized using at most 2 to 5 bits to control the phase shifts.[1] Moreover, practical phased arrays suffer from hardware imperfections [38] which have little impact on the main lobe but can limit the ability to null [38, 43]. For example, in an array with 8 antennas, if the phase on one antenna is off by 5°, the received signal along the main lobe degrades by only 0.004 dB whereas the interference signal along a null increases by 10 dB as we describe in more details in section 4.1.

Furthermore, unlike the main lobe which is naturally wide and, hence, can tolerate small errors in the direction of communication, nulls are narrow as shown in Fig. 2. As a result, any small error in the direction of the interferer will misalign the null and prevent us from effectively eliminating interference. To address this, we must create wider nulls rather than point nulls as shown in Fig. 1(c). In addition, in dense networks, we would need to null multiple different directions to account for multiple interferers or multipath reflections. Creating multiple nulls and wider nulls impose even more requirements that are hard to meet given the constraints and hardware imperfections of practical phased arrays.

Due to the above challenges, past work on null steering remains simulation based [5, 32, 53] and has not been implemented on practical mmWave phased arrays. Furthermore, most past work focuses on creating a single point null and none of the past work accounts for hardware imperfections.

This paper presents Nulli-Fi, the first practical mmWave null steering system that is able to null interference on commercial off-the-shelf phased arrays while preserving the main lobe. Nulli-Fi addresses the above practical challenges by combining a new theoretically optimal algorithm that accounts for limitations in practical phased arrays with a novel discrete optimization framework that overcomes hardware imperfections and enables multiple and wider nulls.

Nulli-Fi's optimal algorithm is able to create a single null within the constraints of practical phased arrays. To understand how this algorithm works, consider the example shown in Fig. 2. The goal is to have the main lobe at 75° and create a null at 100°. Each vector in Fig. 2(d)-(g) represents the signal received on a given antenna element. For signals received along 75°, Nulli-Fi sets the phase shifters to rotate the phase of each of these signals to sum up constructively as shown in Fig. 2(d). For signals received along 100°, the signals will have different phases and the vectors will sum up to some vector $\vec{P}$ as shown in Fig. 2(f). Our goal is to rotate these vectors by changing the phase on the phase shifters in order to null $\vec{P}$ while preserving the main lobe. To do so, Nulli-Fi restricts further phase-shifts on each antenna to a limited range. For example, if we restrict it to $\pm 15°$ on all antennas, the main lobe does not change by more than 0.3 dB, as shown in Fig. 2(e). Nulli-Fi then leverages the insight that the vectors are symmetric around $\vec{P}$ as shown in Fig 2(f).[2] By rotating pairs of symmetric vectors towards $-\vec{P}$, as shown in Fig 2(g), we reduce the amplitude of $\vec{P}$. We iteratively rotate the vectors until we null $\vec{P}$ or achieve the best possible reduction which we prove is optimal given the restrictions on the phase shifts.

The above algorithm provides a simple, optimal way to create nulls under limited phase control but it does not account for hardware imperfections, nor can it create nulls in more than one direction. To address this, we introduce a discrete optimization framework customized to null steering. The framework is inspired by genetic algorithms which have proven effective in discrete optimizations [19, 60]. However, genetic algorithms are very slow and can take thousands of iterations to converge [60] which prevents practical realtime null steering as we discuss in detail in section 7. Like many other optimization techniques, the initialization and stopping criterion are among the most contributing factors to the algorithm's convergence speed [48, 62]. To address this, Nulli-Fi uses the solution to its optimal algorithm. First, it initializes the optimization framework using the solution from the above algorithm which gives Nulli-Fi a significant head-start and helps it converge faster as we show in section 6. Second, since Nulli-Fi's algorithm is optimal, it can serve as a stopping criterion to the optimization framework (i.e., the algorithm knows if it has reached a reasonable solution). Combining the two methods gives Nulli-Fi a powerful framework that is both fast and is able to handle hardware imperfections.

Finally, to enable a practical system, Nulli-Fi develops a

[1] For example, 802.11ad compliant consumer-grade devices use only 2 bit phase shifters, i.e. we can set the phase only to 0°, 90°, 180°, or 270°.

[2] We prove this in lemma 4.2 to be true for any directions of the main lobe and the null for even number of antennas.

Past Work	Analog Beamforming?	Phase only?	Discrete Phase?	Implemented?	HW Imperfections?	Wide Nulls?
[49]	×	×	×	×	×	×
[52]	×	✓	×	×	×	×
[6, 25, 29, 53]	✓	×	×	×	×	✓
[5, 9, 30, 58]	✓	×	×	×	×	×
[32]	✓	×	✓ (1° Res.)	×	×	×
[12, 21, 39, 50, 54, 57]	✓	✓	×	×	×	×
[22, 23, 35]	✓	✓	✓ (6 bits)	×	×	×
[8]	✓	✓	✓ (9 bits)	✓ (at 4.5 GHz)	×	×
[15]	✓	✓	×	✓ (at 2.5 GHz)	×	×
Nulli-Fi	✓	✓	✓ (2-4 bits)	✓ (at 60 GHz)	✓	✓

Table 1: Summary of Related Work on Phased Array Nulling

fast null steering protocol that is able to quickly find the direction in which to create a null whenever a new unforeseen interferer appears. The protocol leverages the intuition that the interferer direction is more likely to be at the large side-lobes shown in Fig. 2(b). Hence, instead of searching all possible directions, Nulli-Fi starts with a large side-lobe where it creates a wide null and iterates through the side-lobes until the interferer is nulled.

We have implemented and extensively evaluated Nulli-Fi using commercial 60 GHz, 16 element phased arrays transmitting IEEE 802.11ad packets [33]. Our results show that for 4 bit phase shifters, Nulli-Fi is able to create 3° narrow nulls that suppress interference by 18 dB and 10° wide nulls that supress interference by 10.5 dB while maintaining the main lobe within 1 dB. For 2 bit phase shifters, Nulli-Fi is still able to null interference by 12.6 dB. Nulli-Fi is also able null up to 5 different directions. We further compare Nulli-Fi with past null forming algorithms and demonstrate up to 10 dB better nulling and 37× faster convergence. We also evaluate NullFi's fast null steering protocol on top of the mm-Flex platform [33] to show that Nulli-Fi can find the direction of an unknown interference and null it within 290 ns. Finally, to demonstrate the effectiveness of Nulli-Fi in dense mmWave networks, we compare Nulli-Fi to past work that leverages the directionality of mmWave radios to enable many concurrent transmissions [27]. By nulling interference from side lobes, Nulli-Fi is able to achieve 2.6× higher data rate when 10 mmWave links (20 nodes) are transmitting concurrently.

Contributions: The paper has the following contributions:

- The paper presents the first practical system that can create nulls on mmWave phased arrays.
- The paper introduces a theoretically optimal algorithm for creating nulls and a novel discrete optimization framework that account for practical challenges in mmWave systems.
- The paper develops a fast null steering protocol to deliver a practical system.
- The system is built and evaluated on real phased arrays to demonstrate significant gains in suppressing interference.
- We have open sourced implementations of our algorithms and baselines on our git repository [36].

2 Related Work

There is a significant literature on millimeter wave beam shaping and steering. Past mmWave systems research, however, has mainly focused on beam alignment, i.e. developing protocols to quickly find the best direction to align the beams of a transmitter and receiver or to switch the beam to a different path to avoid blockage [17, 20, 27, 40, 56, 65, 66]. Some works also explore the problem of beam pattern synthesis [13, 42, 51]. However, these works focus on shaping the main lobe of the beam to achieve good antenna gain along the direction of communication. In contrast, we focus on forming and steering nulls to suppress interference.

Past work on mmWave networks proposes leveraging the directionality of mmWave links to enable dense spatial reuse and maximize the number links that can transmit simultaneously [27, 28]. However, the work shows that side lobe leakage from practical mmWave phased arrays limits the ability to enable spatial reuse. In section 6, we compare with this work to show that Nulli-Fi can enable 2.43× higher throughput than [27] when 10 links are transmitting concurrently. Another work [59] mitigates interference by aligning the natural nulls in the beam pattern toward the interferer. This, however, comes at the cost misaligning the mainlobe [59]. In section 6, we show that this can reduce the SNR by up to 10 dB. In contrast, Nulli-Fi creates new nulls that suppress interference while preserving the main lobe alignment.

Previous work on null forming in phased arrays is simulation based and to the best of our knowledge has not been implemented on practical mmWave phased arrays. Most of the past work ignores many of the practical limitations. Table 1 summarizes past work. Specifically, most methods assume that it is possible to arbitrarily set the phase and amplitude of the complex weights. Others do not require amplitude control but assume phase control is continuous and can be set arbitrary. However, mmWave phased control is highly quantized offering only 2 to 5 bits to control the phase [2, 11, 47]. Two works [49, 52] assume a digital phased array, i.e. each antenna is connected to a district transmitter or receiver and the complex weights can be set arbitrary in digital. Commercial mmWave phased arrays are mostly analog and have a single digital transmitter or receiver as shown in Fig. 2(a) [2–4, 33].

The closest to our work are [22, 23, 35] which use genetic algorithms to create nulls in case of discrete phase only control with 6 bits of quantization. However, these systems are not implemented in practice, ignore hardware imperfections, take many iterations to converge and can only create a point null for which Nulli-Fi has a closed-form solution. In section 6, we implement and compare with these methods to show that even if they account for hardware imperferections, Nulli-Fi still achieves 10 dB better nulling with the same running time, and is 37× faster with the same performance.

Authors in [8, 15] implement nulling on custom built phased arrays. However, they operate in the sub-6 GHz frequency range where it is significantly easier to build phased arrays with flexible control. In particular, [8] works at 4.5 GHz and uses phase shifters with a 9-bit control, i.e. it is possible to set the phase at a resolution of 0.7°. They first solve the nulling problem in the continuous phase domain using gradient descent and then round off the continuous values to the 9-bit discrete space. Millimeter wave phase shifters, however, typically support 2 to 5 bits phase shifters for which the quantization error become too large. In section 6, we implement and compare with this work and show that its performance significantly degrades as the number of bits decreases. Another work [15] operates at 2.4 GHz and use deep neural networks to create the nulls. However, the DNN architecture can only output continuous values and can suffer from over-fitting.[3] In contrast, this paper presents and extensively evaluates a solution that works for highly quantized phase on practical mmWave phased array.

Some works propose changing the positions of the antennas to create nulls in the beam pattern or reduce the side lobes [7, 24, 26, 31]. However, these techniques require new custom built hardware and are only suitable only for static applications with a fixed beam pattern and null locations. Finally, there is a large body of work that proposes interference nulling using MIMO techniques at sub 6 GHz frequencies [16, 18, 34, 37, 45]. These works are complementary to Nulli-Fi as they require multiple digital transmitters or receivers to perform digital beamforming and set arbitrary complex weights in digital to null the signals.

3 Primer

In this section, we provide a primer on phased arrays as well as genetic algorithms on which we base our optimization.

1. Phased Arrays: In analog phased arrays, an array of antennas is connected to a single transmitter or receiver through a single chain. The signal on each antenna n is multiplied with a complex weight $a_n = |a_n|e^{j\alpha_n}$ as shown in Fig. 2(a). By changing these weights, we can change the beam pattern and steer the main lobe of the beam in any direction. The beam

[3] Specifically, the paper mostly provides simulation results and only shows three examples of nulls created on real hardware.

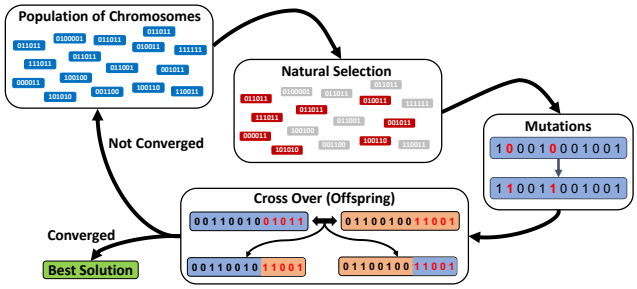


Figure 3: Overview of genetic algorithm

pattern along a direction ϕ can be written as:

$$P(\phi) = \sum_{n=0}^{N-1} a_n e^{2\pi j \frac{d}{\lambda} n cos(\phi)} \quad (1)$$

where N is the number of antennas, λ is the wavelength of the signal, and d is the separation between adjacent antennas. We can steer the main lobe towards the direction ϕ by setting the complex weights to $a_n = e^{-2\pi j \frac{d}{\lambda} n cos(\phi)}$ which will cause the signals coming from direction ϕ to sum up constructively. For example, by setting $\phi = 75°$, we get the beam pattern shown in Fig. 2(b). The beam pattern exhibits natural nulls where $P(\phi) = 0$ and no signal is received along that direction. In practice, however, such perfect nulls are not possible. Hence, we define a null as a point in the beam pattern where $P(\phi)$ is extremely small (e.g. −25 dB relative to the main lobe). The deeper the null, the more effective it is at suppressing interference. Our goal is to find a setting of the complex weights to create a null along a certain angle ϕ_{null} while maintaining the amplitude level of the pattern at $\phi_{\text{main lobe}}$.

If we are able to control both amplitude and phase of the complex weights in a continuous manner, then we can easily create any beam pattern. In particular, we can transform Eq. 1 into a Fourier Transform by setting $f = -d/\lambda \cos(\phi)$. We can then construct any desired pattern and take its inverse Fourier transform to find the set of complex weights that we should use. Most practical phased arrays, however, do not support controlling the amplitude of the complex weights especially since modifying the phase is sufficient to steer the main lobe of the beam. These phased arrays use a component called a phase shifter to shift the phase of the signal on each antenna element. Hence, the problem is restricted to having $|a_n| = 1$, i.e. $a_n = e^{j\alpha_n}$. Unfortunately, the problem becomes even harder when we are limited to a quantized set of phase shifts, especially when the number of control bits used to set the phase shifter is small as the problem becomes non-convex and the search space is exponentially large. For example, for a 16 element array, and 4 bits (= 16 values) of resolution in phase-shifters, we get $16^{16} \approx 1.8\text{e}19$ possible patterns.

2. Genetic Algorithms: Genetic Algorithms (GAs) are a family of evolution-inspired algorithms designed to solve optimization problems. They are particularly useful when the search space is discrete and has many local maxima [60]. A

high-level overview of the algorithm structure is depicted in Fig. 3. The algorithm starts by considering a set of *initial chromosomes* referred to as the *population*. Each chromosome represents one possible solution of the problem e.g. a setting of complex weights $(a_0, a_1, \cdots, a_{n-1})$. The first stage of the algorithm is *natural selection* where the chromosomes are ranked using a fitness function that evaluates how well each chromosome solves the problem e.g. how good of a null it creates. Fraction of chromosomes that are most fit to solve the problem are then selected and the rest are discarded. The remaining chromosomes give rise to new potentially fitter, chromosomes, which repopulate the population via *mutation* and *crossover*. In *mutation*, random bits used to represent the chromosomes are flipped to create new chromosomes whereas in *crossover*, two random parents give birth to two new chromosomes as shown in Fig. 3. Once the population reaches its original size, the fitness of the chromosomes is re-evaluated and the best chromosome is selected. The entire process keeps repeating until the algorithm converges, i.e. reaches some stopping criteria. While genetic algorithms work surprisingly well, they are completely arbitrary and do not exploit the underlying structure of the problem. As a result they take a long time to converge and can give sub-optimal results. Nulli-Fi builds on the high-level structure of such algorithms to design a new optimization framework customized to the problem of null steering.

4 Nulli-Fi

4.1 Nulling Algorithm

Assumptions: To begin, we state the set of assumptions under which we optimally solve the nulling problem. We will assume that the number of antenna elements, N, is even, and that the physical distance of adjacent antenna elements is $d \leq \lambda/2$, where λ is the wavelength. We further assume that we only have phase control over antenna elements, and before nulling, all the antenna elements are beamforming towards some direction ϕ_0, i.e. the phase shifts $\alpha_n = -2\pi n d/\lambda \cos\phi_0$ as descirbed in Eq. 1 [4].

Preserving the Main Lobe: In order to preserve the main lobe of the beam directed towards ϕ_0, we limit any additional phase-shifts on each antenna element to $\pm\alpha^*$, i.e. $|\Delta\alpha_n| \leq \alpha^*$ for all n. We show that this limits the loss in the main lobe to at most $sin^2(\alpha^*)$. In particular, we prove the following lemma in Appendix A.5:

Lemma 4.1 *If $\alpha^* \leq 90°$, a maximum phase shift restricted to $\pm\alpha^*$ for each antenna element will result in a loss of at most $\sin^2(\alpha^*)$ in the main lobe.*

This would mean that for $\alpha^* = 15°$ (or $30°$ of freedom), the main lobe changes by at most 0.3 dB.

[4] In a discrete phase scenario, aligning towards any angle ϕ_0 is not possible and we must set all elements to the closest discrete value to ϕ_0.

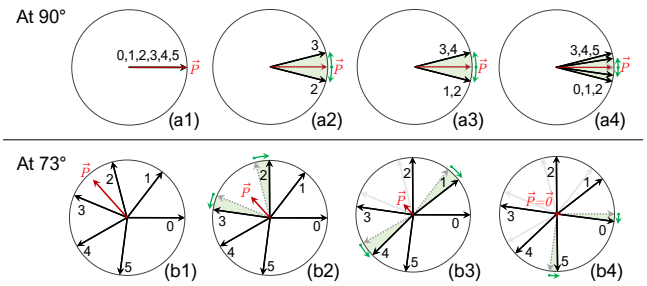


Figure 4: Example Nulli-Fi's nulling algorithm for $N = 6$ antenna elements with the main lobe at $90°$ an nulling at $73°$.

Problem Formulation: Given the restrictions on the phase shifts to preserve the main lobe and the above assumptions, our problem becomes: *Given an angle ϕ, find a set of additional phase-shifts $\Delta\alpha_n$, such that $|P(\phi)|$ is zero (or as close to zero as possible), subject to $|\Delta\alpha_n| < \alpha^*$.*

Algorithm: Our algorithm works by representing the signal on each antenna as a vector in the complex plane. This representation is particularly useful since applying a phase shift is equivalent to rotating these vectors. Thus, our goal is to rotate these vectors to null the signal in the direction of ϕ. To better understand how this works, consider the example shown in Fig. 4. In this example, we have $N = 6$ antenna elements beamforming towards $\phi_0 = 90°$. The vectors $\vec{v_n}$ representing the signal on each element are indexed by: $0, 1, \cdots, 5$, and our goal is to create a null at $\phi = 73°$.

Initially, the vectors are aligned to sum up constructively to $\vec{P_{\phi_0}}$ along $90°$ as shown in Fig. 4(a1). [5] However, they are aligned differently along $73°$ and sum up to $\vec{P_\phi}$ as the signals come with a different phase at that direction as shown in Fig. 4(b1). To create a null along $73°$, we will rotate each vector by an additional $\Delta\alpha_n$ to minimize the $\vec{P_\phi}$. The restriction $|\Delta\alpha_n| \leq \alpha^*$ will ensure that $\vec{P_{\phi_0}}$ along the main lobe is preserved as shown in Fig. 4(a2–a4). However, it will prevent us from arbitrarily rotating the vectors along $73°$. To address this, we leverage the following key observation: *At any direction ϕ, all the vectors summing up to the pattern $\vec{P_\phi}$ come in pairs symmetrically located around the pattern*. For example, in Fig. 4(b1), the following pairs: $\{0,5\}$, $\{1,4\}$ and $\{2,3\}$ are symmetrically located around $\vec{P_\phi}$.

The following lemma formalizes this observation. The proof of the lemma can be found in Appendix A.5.

Lemma 4.2 *At any direction ϕ, if $\Delta\alpha_n = 0$ for all n, then $\vec{v_n}$ and $\vec{v}_{N-1-n}$ are symmetrical around $\vec{P_\phi}$ for all n. That is, $\frac{1}{2}(\angle\vec{v_n} + \angle\vec{v}_{N-1-n})$ is the same as $\angle\vec{P_\phi}$ or $\angle\vec{P_\phi} + \pi$.*

Given this observation, the algorithm proceeds as follows. Choose a pair of symmetrical vectors around the pattern $\vec{P_\phi}$ and symmetrically rotate them towards $-\vec{P_\phi}$ as much as possible (i.e. until α^* degrees, or until a null is achieved). This

[5] In fact, $\vec{P} = 6e^{j0}$ but we have downscaled it by 6 for better visualization.

will reduce the beam pattern amplitude along ϕ as shown in Fig. 4 (b2) but it will *not* change its angle. This means that all the vectors remain symmetrical around the $\vec{P}_\phi$. If a null is achieved, we stop. If not, we repeat with another symmetrical pair as shown in Fig. 4 (b3,b4). Note that the same rotations are also applied at the main lobe in Fig. 4 (a2-a4). While these rotations result in a null at 73°, they cause only a 0.2 dB loss at 90°.

A pseudocode of the algorithm can be found in Alg. 1 in Appendix A.1. We also prove the following theorem regarding the optimality of our algorithm in Appendix A.5.

Theorem 4.3 *Given the constraint $|\Delta\alpha_n| < \alpha^*$, Alg. 1 gives the best nulling performance at any angle ϕ.*

It is worth noting that given the constraints, it is not always possible to achieve a perfect null i.e. $\vec{P}_\phi = \vec{0}$. In such cases, the above algorithm yields the deepest possible null. This also allows the algorithm to identify directions that can be perfectly nulled from those that cannot. In Appendix A.4, we provide further analysis and closed form solutions for the bounds of achievable nulling performance as a function of the direction of the null.

4.2 Optimization Framework

In this section, we show how to account for hardware imperfections and achieve multiple and wider nulls. We extend our definition of a null to be an interval 2β degrees wide around ϕ i.e., $[\phi-\beta, \phi+\beta]$ where the magnitude of the beam pattern is lower than a certain threshold. The input to our optimization are multiple such intervals $([\phi_i-\beta_i, \phi_i+\beta_i])$ where we wish to null interference. A pseudocode of our optimization framework can be found in Alg. 3.

Encoding: We will encode the solution i.e. the setting of the phase shifts α_n into chromosomes that form the basis of the genetic algorithm. Suppose the phase shifts are quantized using q bits, then each α_n can be represented as a bit string $(b_{n,1}, \cdots, b_{n,q})$ where $b_{n,i}$ is the i^{th} most significant bit of α_n. A chromosome A can then be encoded as a concatenation of the N binary representations of the phase shifts: $A = (b_{0,1}, \cdots, b_{0,q}, b_{1,1}, \cdots, b_{1,q}, b_{N-1,1}, \cdots, b_{N-1,q})$. We define P_A as the beam pattern associated with chromosome A.

***Initialization*:** While genetic algorithms generally start from a set of randomly generated chromosomes, we use the output of Alg. 1 to initialize our genetic algorithm. Specifically, for each null region $([\phi_i-\beta_i, \phi_i+\beta_i])$, we run Alg. 1 and find the optimal setting of α_n to create a null along ϕ_i. Each solution will give us a single initial chromosome. We then slightly perturb the values of the phase shifts to create a larger population of initial chromosomes. This dramatically improves the optimization's performance as we show in section 6.2.

***Fitness function* $F(A)$:** This function evaluates the performance of any given chromosome A. In our problem setup, we define the fitness function as

$$F(A) = \min_{\substack{i=\{1,\ldots,L\} \\ \phi \in [\phi_i-\beta_i, \phi_i+\beta_i]}} -10\log_{10}\left(|P_A(\phi)|^2\right),$$

where $P_A(\phi)$ can be calculated from Eq. 1 by setting the complex weights to $e^{j\alpha_n}$. This fitness function $F(A)$ optimizes for the *worst* nulling performance in dB across all the regions we wish to null. In particular, the min point of $-10\log_{10}\left(|P_A(\phi)|^2\right)$ is the max point in $|P_A(\phi)|^2$ which is the least nulled point. Hence, the fittest chromosome, $A^* = \arg\max_A F(A)$, will give the best nulling performance across all directions since we optimized for the worst case.

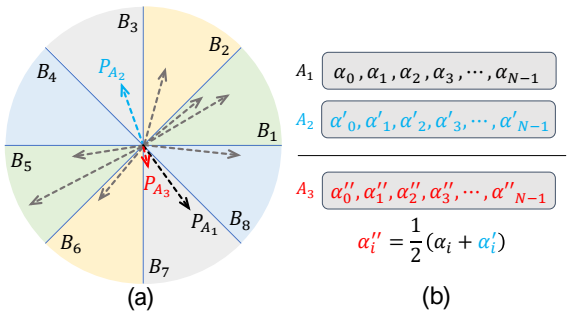


Figure 5: Nulli-Fi's crossover operation using buckets

***Natural Selection*:** At each iteration, we evaluate the fitness function for every chromosome and keep the ones with the best performance.In our implementation, we typically keep the top 50% of the chromosomes.

Cross-over. Recall from section 3, this operation is meant to combine two parent chromosomes A_1 and A_2, to give birth to a new, potentially fitter chromosome, A_3. Typically, the two parents A_1 and A_2 are chosen randomly. However, Nulli-Fi employs a more intelligent selection criteria. For simplicity, let us consider a single null point and use the same vector representation we used in section 4.1 to explain Nulli-Fi's cross-over operation.

To begin, we first group chromosomes into different buckets $1, \cdots, 2B$. Bucket i contains all chromosomes A with $(i-1)\frac{\pi}{B} \leq \angle\vec{P}_A < i\frac{\pi}{B}$. Fig. 5 (a) shows an example of these buckets for $B=4$, where buckets on the opposite sides of each other have the same color. In our cross-over operation, two parents A_1 and A_2 are then chosen at random, under the constraint that $\vec{P}_{A_1}$ and $\vec{P}_{A_2}$ are in opposing buckets (for example, B_3 and B_7). Then, a new chromosome A_3 is created by averaging the phase shifts of A_1 and A_2, as shown in Fig. 5 (b). The intuition behind this is that by taking the average phase shift of the two parents, the new chromosome will approximately have a pattern vector equal to the sum of its parents. Since the parents come from opposing buckets, the summation of their patterns will likely result in a smaller vector. This is depicted in Fig. 5 (a) where the red vector corresponding to the child chromosome A_3, is smaller than the pattern of either parent (depicted black and blue vectors). By exploiting the structure of the problem, Nulli-Fi is able to quickly generate fitter

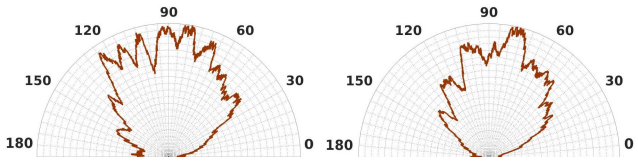


Figure 6: Non uniform radiation patterns of antenna elements

chromosomes with smaller $|P_A(\phi)|$ i.e. deeper nulls which improves the results as we show in section 6.2.

Mutation. In this step, we randomly flip bits in the parent chromosome with some probability to give rise to a new chromosome. Note that we preserve the current best chromosome A^* which remains unchanged during mutation.

Convergence. The algorithm converges once the best performing chromosome reaches a fitness threshold, or a maximum number of iterations has been reached. In the case of a single null, this threshold is directly governed by the output of Alg. 1. For multiple nulls, as one would expect, the performance usually does not reach the theoretical performance of a single null. In our implementation, we reduce the threshold by around 1 dB for every extra null region.

Preserving the main lobe. Similar to the optimal algorithm, we maintain $|\Delta\alpha_n| \leq \alpha^*$ to preserve the main lobe. This can be done by simply fixing the $q - \log_2(\pi/\alpha^*)$ most significant bits of α_n and not changing them throughout the optimization. However, in cases where q is very small, e.g. 2 bit phase shifters, such an approach does not hold. To address this, Nulli-Fi sets aside a subset of the antenna elements and does not change their phase shift throughout the entire optimization. This subset will contribute to the main lobe whereas the remaining antennas will contribute to create the null. Nulli-Fi dynamically chooses the antenna elements that are contributing the most to the main lobe to be in this subset and allows phase shifts for the ones that are contributing the least to the main lobe. A pseudocode for this process can be found in Alg. 2 in section A.1 of the appendix.

Accounting for Hardware Imperfections: There are two types of hardware imperfections: (1) phase offsets due to different wire lengths or paths that the signals traverse, and (2) non-uniform antenna element radiation patterns. In particular, the signal on each antenna incurs an additional δ_n and signals coming from a direction ϕ incur an additional attenuation of $R_n(\phi)$. Fig. 6 shows the radiation patterns of two antenna elements on our hardware setup described in section 5. As can be seen, antennas do not receive the signal uniformly across all directions. While these factors do not severely affect the quality of the main lobe, they have a more significant impact on the nulling performance of the phased array, as we show in section 6. This is because the beam pattern computed using Eq. 1 and used for evaluating the fitness function is no longer valid in practice. We can measure these imperfections using a simple calibration procedure outlined in detail in appendix A.2. Once measured we can modify Eq. 1

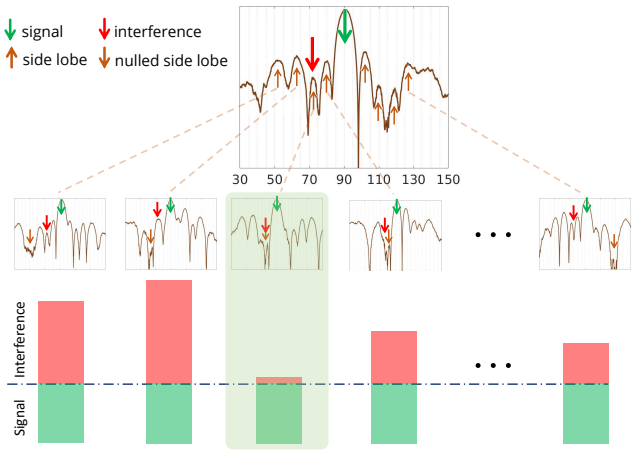


Figure 7: Illustration of Nulli-Fi's nulling alignment and interference suppression.

as follows:

$$P(\phi) = \sum_{n=0}^{N-1} a_n R_n(\phi) e^{j\delta_n} e^{-2\pi j \frac{d}{\lambda} n cos(\phi)}, \quad (2)$$

We observe these imperfections to be stable and, hence, can be measured once. By modifying the fitness function to account for these hardware imperfections, we can generate beam patterns that achieve good nulling performance in practice.

4.3 Fast Null Steering Protocol

Now that we have a framework to form nulls at any desired direction, we need to find a practical way to align and suppress nulls in a real network. In this section, we present a simple yet fast and practical protocol to do so. The protocol finds and suppresses interferers in succession, by enforcing wide nulls at the high-level side-lobes of the pattern.

We begin with a simple example, where there is only one interferer. Consider the pattern in Fig. 7. As can be seen, the pattern has a number of significant side lobes, denoted by upwards brown arrows, which are the most likely to receive interference from other links in the network. Nulli-Fi finds these side-lobes, and computes corresponding patterns that have nulls at each side lobe as shown on the second row in Fig. 7, while keeping the main lobe. The hardware then quickly sweeps through these patterns, computing the Received Signal Strength (RSS) corresponding to each pattern. If the RSS drops for one of these patterns, it means that the interference was suppressed. This way, we can eliminate the interferer. To make the algorithm even faster, Nulli-Fi checks the SINR value at after each beam switch, and stops if the SINR is within a threshold of its original value.

Following this example in case of multiple interferers, we first suppress the one with the highest power. Once this interferer is nulled, we keep a null at its direction at all times,

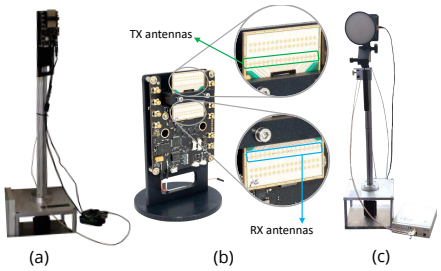


Figure 8: Hardware used in Nulli-fi.

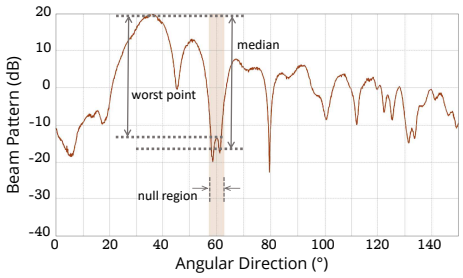


Figure 9: Defining the evaluation metrics.

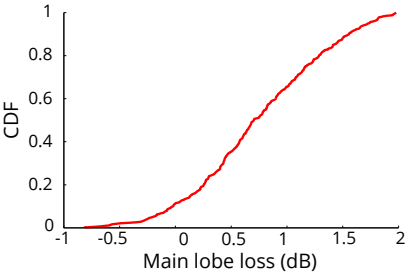


Figure 10: Main lobe loss in Nulli-Fi.

and search for the next interferer with the highest power. We repeat this process until all interferers are suppressed. We note that since our protocol only looks for interferences at high-level side lobes of the pattern, it will be much faster than a full scan, while remaining effective in improving the SINR, as we show in section 6.4.

5 Implementation

Nulli-Fi's Setup. Nulli-Fi is implemented using the off-the-shelf Sivers IMA EVK06002 platform [3], equipped with a 60 GHz 16-element linear phased array shown in Fig. 8(b). In order to measure the beam patterns, we mount the phased array radios on a steerable platform controlled through an Arduino as shown in Fig. 8(a). Our testbed also includes a 60 GHz Pasternack PEM009-KIT [1] equipped with a directional 3-degree horn antenna (Fig. 8(c)) which we use to transmit signals in order to measure the generated beam patterns. All hardware devices were connected to a machine running Ubuntu 18.04 through USRP N210 software defined radios. The center frequency in all experiments was 60.48 GHz. We run our experiments in 4 different rooms in 8 different locations, and in each location, we test 125 distinct combinations of directions of communication and interference. The EVK06002 platform offers flexible phase control for each of the antenna element weights which allows us to experiment using different number of bits. We evaluate Nulli-Fi using at most 4 bits of phase resolution, i.e. 16 distinct phase shift values per antenna element. We also show Nulli-Fi's performance for more coarse-grained control on phase, specifically 2 bit and 3 bit phase resolution. We also calibrate the array as described in Appendix A.2.

Nulli-Fi + mm-Flex Setup. We also implemented Nulli-Fi on top of the mm-Flex platform [33], to evaluate our null steering protocol. We transmitted IEEE 802.11ad control frames, where we use 10 Golay sequences to switch beam patterns. Our setup can switch between beam patterns once every 54.5 nanoseconds, during which we are able to measure RSS values corresponding to that pattern. This way, sweeping through 10 beam patterns takes less than 0.55μs. Further details of this setup can be found in Appendix A.3.

6 Results

6.1 Evaluation Metrics

We start by describing the evaluation metrics used to quantify Nulli-Fi's performance. Fig. 9 illustrates some of the metrics on an example beam pattern created by Nulli-Fi.

- *Null Width:* Since we create wide nulls, we define the null width as the region of directions nulled in the beam pattern.
- *Worst-point nulling performance:* Minimum amount of nulling in the target null region, measured as the difference between the peak of the main lobe and maximum point in the null region as shown in Fig. 9.
- *Median nulling performance:* Median amount of nulling in the target nulling region measured as the difference between the peak of the main lobe and median point in the null region.
- *SINR Gain:* Gain in Signal-to-Interfernce plus Noise Ratio before and after nulling the interferer.
- *Main Lobe Loss:* Loss in the main lobe power compared to a beam pattern without the imposed nulls.
- *Number of Nulls:* Number of different directions in which nulls are created.
- *Quantization Level:* Number of control bits used to set the phase on the phase shifters.
- *Number Iterations:* Number of iterations it takes for the optimization to converge.
- *Null Steering Latency:* Time it takes to steer the null towards an interferer once the interferer appears.

6.2 Baselines

We compare Nulli-Fi to the following baselines:

(1) Quantize-Continuous [8] – This baseline solves the problem in the continuous domain and then quantizes the phase solution to the nearest available discrete phase values.

(2) Genetic-Algorithms [22] – We compare Nulli-Fi to past work that uses genetic algorithms to create nulls.

(3) Shift-Pattern [59] – This baseline steers the main lobe a bit away from the direction of communication in order to align the natural nulls in the signal with the interferer.

(4) BounceNet [27] – This work aims to enable dense spatial reuse in mmWave networks by leveraging the directionality of mmWave beam patterns.

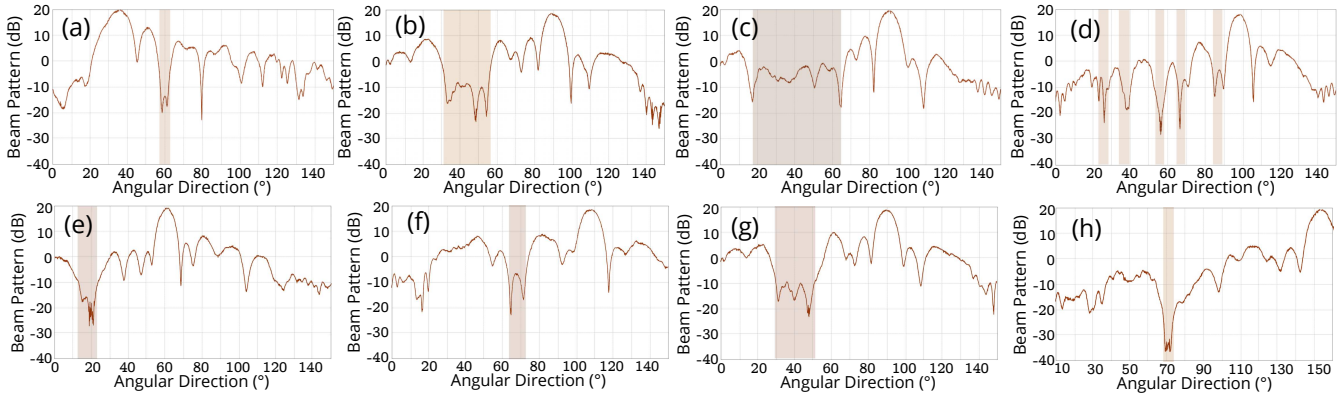
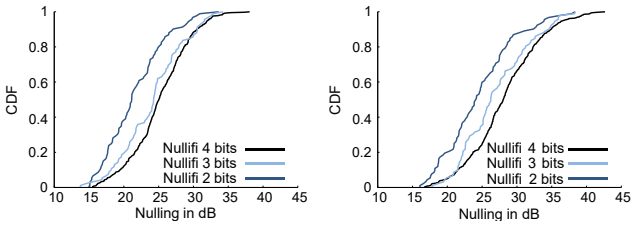


Figure 11: Examples of nulls created in hardware with different null with, number of nulls, direction of the null, and direction of the main lobe.



(a) worst-point performance (b) median performance

Figure 12: Nulli-Fi's performance using fewer bits.

6.3 Nulling Performance

We start by evaluating Nulli-Fi's ability to create nulls and compare it with past work.

1. Nulli-Fi's Nulling Performance: Fig. 11 shows a few examples of beam patterns with nulls created by Nulli-Fi on our phased array for different null directions, main lobe directions, null widths, and number of nulls using 4-bit phase shifters. As can be seen in Fig. 11(e, h), Nulli-Fi can create nulls as deep as -20 dB and -35 dB (40 dB and 55 dB below the main lobe respectively). Nulli-Fi can also create nulls that are as wide as 20° while maintaining a median nulling performance that is 20 to 25 dB below the main lobe as shown in Fig. 11(b, g). It can create up to 5 different nulls as shown in Fig. 11(d).

Fig. 12 shows a CDF of the median and worst-point nulling performance in more than 1000 experiments. Nulli-Fi's 50^{th} percentile is about 29 dB for the median nulling and 25 dB for the worst-point nulling. Since commercial 802.11ad hardware today, like laptops and tablets, comes equipped with only 2 bit phase control in the phased arrays [2, 11], we evaluate Nulli-Fi's performance using fewer bits of phase resolution. Fig. 12 also plots the CDF of the nulling performance with Nulli-Fi using 2 and 3 bits of phase control. While the nulling performance degrades with fewer bits, Nulli-Fi is still able to achieve a 24.1 dB median and 21.1 dB worst-point performance using only 2 bits, and 26.2 dB median and 24.3 dB worst-point performance using 3 bits of phase resolution.

Finally, we measure the main lobe loss suffered due to creating nulls, and plot the empirical CDF in Fig. 10. As can be seen, the median and the 90^{th} percentile values of the main lobe power loss are only 0.58 dB and 1.46 dB respectively, demonstrating Nulli-Fi's ability to preserve the main lobe while creating nulls.

2. Nulling Performance vs. Null Width: There is a natural trade-off between the width of the null created and its median (or worst-point) performance. To examine this, we evaluate Nulli-Fi's ability to create very narrow nulls like the ones shown in Fig. 11 (a, h) as well as very wide null regions like the ones shown in Fig. 11 (b, c). We run experiments where we create nulls of different widths ranging from 1° to 50°, and plot the median and worst-point performance in Fig. 13(a).

While the median nulling performance remains more than 25 dB even for nulls as wide as 50°, the worst-point performance deteriorates much more quickly. This point becomes clear when we consider the fact that the total radiated power in the beam pattern has to be conserved, implying that nulling one region will cause other regions to have an amplification in power. Therefore, while it may be possible to keep the median point in the target null region low for wide nulls, it becomes increasingly difficult to ensure that the worst-point in the target null region remains low as well.

It is worth noting that such overly wide nulls might not be needed in practice and nulls with width of 5° to 10° might be more than enough to account for inaccuracy in the estimating the direction of interferer. One could, however, use very wide null to preemptively supress less powerful interferers.

3. Nulling Performance vs. Number of Nulls: As mentioned previously, in practical networks there could be multiple sources of interference (separate signals or multipath), each occurring at a different angle. Thus, we test Nulli-Fi's ability to create l simultaneous null regions for $1 \leq l \leq 5$. We run 200 experiments for each l by randomly assigning the null regions. We constrain all null regions to be at most 10° wide, and in the case of multiple nulls, any two null regions should be at least 10° apart (otherwise, we observe that the two nulls

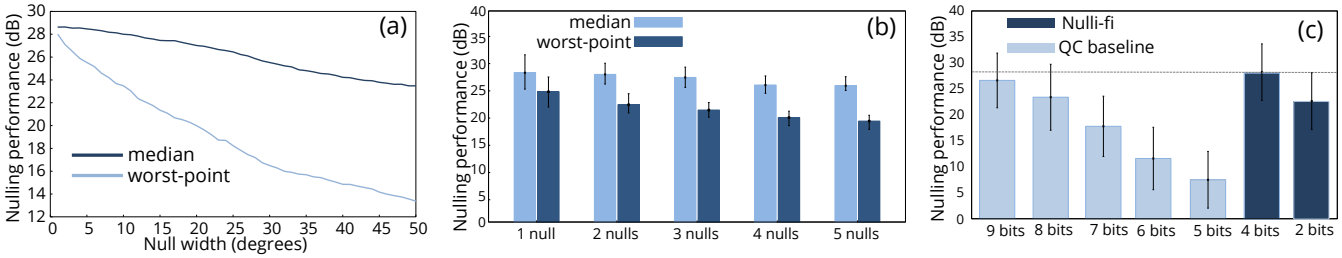


Figure 13: (a) Null performance versus width of the null. (b) Null performance versus number of nulls. (c) Nulli-Fi verus baseline as a function of number of quantization bits.

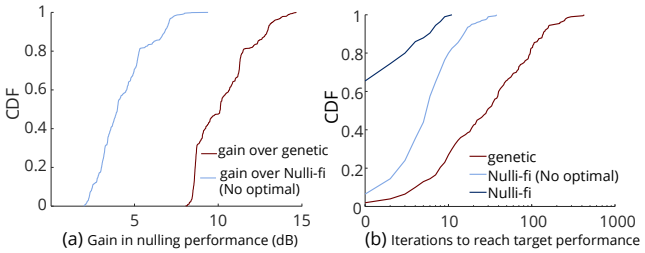


Figure 14: performance of Nulli-Fi's algorithm against baselines

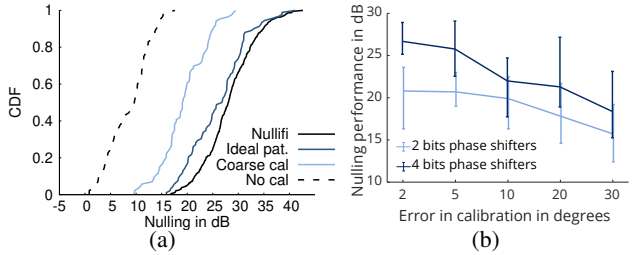


Figure 15: Importance of accounting for hardware imperfections and sensitivity to calibration errors.

merge into one wider null). Fig. 13(b) shows the median and worst-point nulling performance for different numbers of null regions. Note that for multiple null regions, we present the median and worst-point performance numbers for the poorest performing null in the beam pattern. As can be seen in Fig. 13(b), even with 5 null regions, Nulli-Fi is able to achieve 25.8 dB median and 19.1 dB worst-point performance. Fig. 11(d) shows an example of 5 null regions generated by Nulli-Fi.

4. Nulling Performance vs. Baselines: We run more than 1000 experiments for creating nulls at different angles, with different main lobe directions and null widths. Our nulling angles and main lobe directions range from the 30° to 150° region, and null widths range from 5° to 20° width. In all of the experiments, the target null region does overlap with the $\pm 10°$ region around the main lobe direction. In these experiments, we focus on the performance for creating a single null. In Fig. 13(c), we compare Nulli-Fi's performance against the *Quantize-Continuous* baseline. The continuous solutions are quantized to $b = 5, 6, \cdots, 9$ bits of phase resolution, whereas Nulli-Fi is implemented on real hardware with 4 bits of phase resolution. Nulli-Fi's performance exceeds *Quantize-Continuous* even with 9 bits of phase resolution compared to Nulli-Fi's 4 bits. This shows that simply quantizing the continuous phase solution (especially quantizing to less than 7 bits) does not work for practical phased arrays.

We also compare Nulli-Fi with genetic algorithm [22] (*Genetic*) as well as with Nulli-Fi's optimization framework without initializing it with the solution of our optimal algorithm (*Nulli-Fi No optimal*). We run experiments where each algorithm is required to create single nulls at 200 different directions. We fix a target nulling performance of -20 dB and record the number of iterations required to achieve the desired nulling. Fig. 14(a) plots a CDF of the No. of iterations, showing that Nulli-Fi converges almost two orders of magnitude faster than *Genetic*. The figure also shows that Nulli-Fi's optimal algorithm enables much faster convergence and in many cases already gives a nulling performance of -20 dB. Hence, Nulli-Fi converges in a single iteration.[6] Moreover, By comparing the 99^{th} percentile of *Genetic* and *Nulli-Fi(No optimal)*, which comprises cases where it is more difficult to create nulls, we can see that Nulli-Fi's novel crossover scheme helps in pushing the algorithm faster towards the desired nulling performance. Next, we fix the number of iterations to 10 for all three algorithms and plot the CDF of the nulling gain achieved by Nulli-Fi over each algorithm in Fig. 14(a). Nulli-Fi achieves a median gain in nulling of 10 dB over *Genetic* and 4 dB over *Nulli-Fi (No optimal)*.

5. Sensitivity to Calibration & Hardware Imperfections To show the significance of accounting for hardware imperfections, we run experiments to evaluate nulling performance using coarse and absent calibration on the phased array front-end. We also run experiments without accounting for non-uniform antenna radiation patterns discussed in Section 4. As mentioned previously, such imperfections have little effect on the location and power of the main lobe, but will lead to significant errors in null forming [43]. Fig. 15a, shows a CDF of the nulling performance. Without accounting for hardware imperfections, the median nulling is only 10 dB which is 17 dB worse than Nulli-Fi. With simple coarse calibration, the performance already improves by 7 dB. The figure also shows that while ignoring the non-uniform radiation patterns is not as severe, it still reduces the median nulling performance by 3 dB compared to Nulli-Fi. Finally, Fig. 15b shows the sensitiv-

[6]The baselines might also converge in a single iteration if the desired null happens to align with a natural null in the beam pattern.

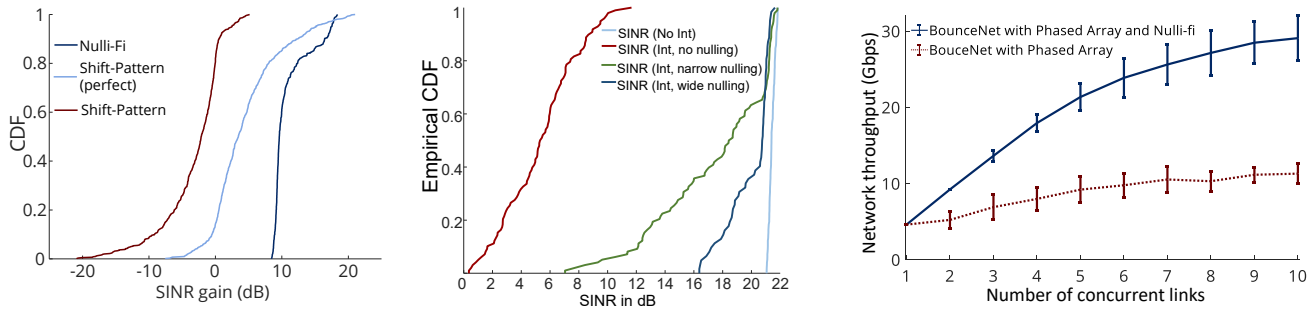


Figure 16: (a) Nulli-Fi's SINR gain over shifting the pattern baseline. (b) Nulli-Fi's ability to steer to and suppress interference. *Int* in the legend indicates the presence of interference. (c) Nulli-Fi's network throughput gains in dense networks

ity of Nulli-Fi's performance to errors in calibration. While the performance degrades as the calibration error increases, the figure shows that Nulli-Fi is robust to calibration errors less than 5° and can still null even if the calibration errors are 30°. It is worth noting that the degradation is less sharp in the case of 2 bit phase shifters. This is likely due to the fact that the phase is highly quantized and hence any calibration error is within the quantization errors.

6.4 Suppressing Interference & Improving Throughput

In this section, we present results for Nulli-Fi's ability to steer the null to suppress interference.

1. Null Steering Algorithm: Here we show the performance of Nulli-Fi's ability to suppress new, unforeseen interferences. To this end, we implemented Nulli-Fi on mm-Flex [33], and we ran close to 700 experiments with different relative locations, power levels for the interference. The experimental setup of this part was explained in section 5.

Fig. 16(b) shows the CDF of Signal to Interference plus Noise (SINR) ratio under four different conditions. As revealed by the figure, throughout all experiments, we set the SINR for when there is no interference to around 21 dB. By introducing the interference (shown by the maroon curve) whose location and power is unknown to the system, the SINR drops to as low as < 1 dB. Then, by running Nulli-Fi's null steering algorithm as described in section 4.3, Nulli-Fi chose and suppressed the side lobes one by one in order to restore the original SINR. The algorithm would stop once it reached within 1 dB of the original SINR, or it suppressed each side-lobe once (up to 10 side lobes).

We did this experiment in two regimes of narrow (2°) and wide (10°) nulls, shown by green and dark blue curves respectively. As seen in both curves, Nulli-Fi can improve the SINR by a median of 13 dB and 15 dB for narrow and wide nulls respectively [7]. We therefore see that Nulli-Fi is able to bring the SINR very close to its original value in the absence of the interferer in all cases. This shows that it is sufficient to look for interference only at the side lobes, as opposed to performing a full scan.

We mention a trade-off between using narrow (2°) and wide (10°) nulls. We expect wide nulls to have a higher chance of capturing the interference, albeit with lower suppressing power as we showed in section 6.3. We see here that this is indeed the case: Compared to narrow nulls, wide nulls have a higher chance of capturing the interference, while narrow nulls suppress the interference better. This is also reflected by the tails of the green and the dark blue curves in Fig. 16 (b). We also compare the runtime of Nulli-Fi's algorithm against the baseline of fully scanning all angles. The numbers are reported using the fast beam switching and RSS measurement technique implemented in [33]. We see that Nulli-Fi's algorithm run on average in 290 nano-seconds, with a standard deviation of 115ns, which is more than 10× faster than a full search scheme, whose average and standard deviation for running time are 3.280 and 1.616 μs, respectively.

Finally, we compare Nulli-Fi's performance in gaining SINR with the Shift-Pattern baseline. We fix the signal and the interference power, and we move the interferer to different angles, and run 100 experiments to measure the gain in SINR. We compare Nulli-Fi with this baseline in two cases. In the first case, *Shift-Pattern (perfect)*, we assume perfect knowledge of the beam pattern, in which case Shift-Pattern chooses the best (deepest) null direction out of all direction within an interval of 10 degrees around the current pattern. We note that although this always reduces the power at the desired null location, it may lead to significant losses in the mainlobe, as we can see in Fig. 16(a). Things get even worse once we use the theoretical beam pattern to predict the optimum shifting amount (*Shift-Pattern*), which almost always results in a loss of SINR, due to inaccuracy of the theoretical beam pattern in predicting the real one. Nulli-Fi, on the other hand, always gives at least 8*dB* improvement in SINR, outperforming both versions of the baselines in almost all cases. This shows that simply shifting the pattern does not work in a practical system, since by shifting towards a null, we also shift the main lobe away from the direction of communication.

2. Throughput in Dense Networks: Fig. 16(c) demonstrates

[7] These experiments were run in the IMDEA networks lab. We found that the SINR gains of Nulli-Fi + mm-Flex is around 3-5 dB larger than Nulli-Fi alone due to a slightly different hardware setup and a lower noise floor

No. of Links	Max Gain	90^{th} Perc. Gain	Median Gain	No. of Links	Max Gain	90^{th} Perc. Gain	Median Gain
1	1×	1×	1×	6	3.60×	2.83×	2.33×
2	2×	2×	1.58×	7	3.50×	2.72×	2.38×
3	3×	3×	1.8×	8	3.38×	2.94×	2.41×
4	4×	2.86×	2.12×	9	2.97×	2.77×	2.44×
5	4.16×	2.81×	2.27×	10	3.09×	2.68×	2.43×

Table 2: Gains in Total Network Data Rate from Nulli-Fi

Nulli-Fi's performance gains in dense networks. To do so, we implement and compare with BounceNet [27] which exploits the directionality of mmWave phased arrays to enable dense spatial resuse. We incorporate Nulli-Fi's nulling into BounceNet. Fig. 16(c) plots the total network data rate as the number of links in the network increases from 1 to 10. We compare Nulli-Fi against a regular phased array testbed using standard codebook-based beam patterns without interference nulling. As seen in the figure, due to significant interference in dense networks caused by side lobe leakages and multipath, a regular phased array equipped testbed can achieve only up to 11.31 Gbps network data rate for 10 links. Nulli-Fi, on the other hand, can effectively null out interference at each link and can increase the total data rate for the same phased array testbed to 29.1 Gbps, providing a gain of 2.6×.

In Table 2, we present further statistics on the gains in total data rate achieved by Nulli-Fi over a regular phased array testbed for different number of links n in the network. For each n we perform 100 different experiments by randomizing the client and AP positions. The result shows that for up to $n = 4$ communication links, Nulli-Fi can achieve the maximum possible gain of $n\times$ over the vanilla phased array testbed. Thus, in certain experiments Nulli-Fi was able to get all 4 links to communicate simultaneously by nulling out interferences, whereas the regular phased arrays were not able to exploit any spatial reuse whatsoever due to side lobe leakages and interference. Note that this gain saturates and begins to fall as the number of links increases due to increased interference. Nonetheless, Nulli-Fi is still able to achieve gains as high as 3.09× in network data rate for 10 links in the network. Table 2 also shows results for 90^{th} percentile and median gains.

7 Discussion and Limitations

In this paper, we introduced novel algorithms that significantly boost the convergence speed and improved the nulling performance compared to past work. Furthermore, the system enabled the first practical implementation of null steering by accounting for hardware restrictions, incorporating hardware imperfections and achieving wide and multiple nulls.

Importance of Convergence Speed: One might wonder, however, why having a faster algorithm is important in practical network deployments. The reason has to do with today's commercial phased array hardware. In particular, the hardware typically stores a codebook of different beam patterns in the on-board memory, and the mmWave radio beams towards different directions by reading the precomputed phase shift values from the codebook. As such, it is not possible to store precomputed beam patterns for all combinations of main-lobes and nulling directions. For instance, if we consider beam patterns with just one null, we would need to store a beam pattern corresponding to each main-lobe direction and each null direction, so a total of 180×180 beam patterns to achieve a null accuracy of 1 degree. This requirement grows exponentially with the number of nulls and would require gigabytes of memory for more than 2 nulls. Compare this to today's millimeter wave phase array that can store 16 to 256 codebooks. Hence, pre-computing and storing the beam patterns is not feasible. This is precisely why it is important to have an efficient algorithm that can converge quickly and compute the required beam patterns in real-time operation. This can allow even further optimization of the beam pattern at run-time which was not possible earlier in the codebook approach. Therefore, the speed of convergence is an important metric in evaluating the different nulling algorithms.

Limitations. We point out a few matters worth considering.

• In this paper, Nulli-Fi enables nulling the interference at the receiver. This is because it is easy for receivers to sense the direction of interference and change their beam pattern to suppress it. That said, there is an opportunity to perform nulling from the transmitter side where the transmitter creates a null in its beam pattern to suppress its own signal in direction of other receivers. This, however, would require an efficient protocol that allows the transmitter to discover the direction of those other receivers at which it is creating interference. Performing nulling from both transmitter and receive side would further improve the performance of the network. However, we leave that for future work.

• Once Nulli-Fi successfully nulls an interferer, it may not sense when it disappears. As a result, if new interferers appear, Nulli-Fi may not know whether to create more nulls or to switch the direction of the null. This can potentially be solved by periodically checking each nulled region for the presence of interference when it is not receiving packets.

• Nulli-Fi's framework is designed for phase shifters that use analog beamforming, which is common for commercial, practical phased arrays. While digital beamforming introduces a substantial overhead in terms of cost and power consumption, the in-between class of hybrid beamforming allows for more flexibility in terms of nulling. Exploring nulling in hybrid beamforming is left for future work.

Acknowledgement

We would like to thank our shepherd Bo Chen as well as the reviewers for their feedback. We would also like to thank Sepehr Madani and Junfeng Guan for their comments. We are also grateful to NSF (award numbers: 1750725, 1824320), Google, Facebook Connectivity Lab, and the Sloan Foundation for partially funding this research.

References

[1] Pasternack pem009-kit. https://www.pasternack.com/60-ghz-development-systems-category.aspx. Accessed: 2020-09-17.

[2] Qualcomm 802.11ad products to lead the wayfor multi-band wi-fi ecosystem. bit.ly/2Fy2CnM. Accessed: 2020-09-17.

[3] Sivers ima evk06002 platform. https://www.siversima.com/product/evk-06002-00/. Accessed: 2020-09-17.

[4] O. Abari, H. Hassanieh, M. Rodreguiz, and D. Katabi. Poster: A millimeter wave software defined radio platform with phased arrays. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 419–420, 2016.

[5] O. P. Acharya, A. Patnaik, and S. N. Sinha. Null steering in failed antenna arrays. *Applied Computational Intelligence and Soft Computing*, 2011:4, 2011.

[6] K. Akdagli. Null steering of linear antenna arrays using a modified tabu search algorithm. *Progress In Electromagnetics Research*, 33:167–182, 2001.

[7] A. Alphones and V. Passoupathi. Null steering in phased arrays by positional perturbations: a genetic algorithm approach. In *Proceedings of International Symposium on Phased Array Systems and Technology*, pages 203–207. IEEE, 1996.

[8] C. Baird and G. Rassweiler. Adaptive sidelobe nulling using digitally controlled phase-shifters. *IEEE Transactions on Antennas and Propagation*, 24(5):638–649, 1976.

[9] G. C. Bower. Simulations of narrow-band phased-array null formation for the ata. *ATA Memo Series*, 37, 2001.

[10] L. Chettri and R. Bera. A comprehensive survey on internet of things (iot) toward 5g wireless systems. *IEEE Internet of Things Journal*, 7(1):16–32, 2019.

[11] N. Choubey and A. Panah. Introducing facebook's new terrestrial connectivity systems-terragraph and project aries. *Facebook Research*, 2016.

[12] D. A. Day. Fast phase-only pattern nulling for large phased array antennas. In *2009 IEEE Radar Conference*, pages 1–4. IEEE, 2009.

[13] D. De Donno, J. Palacios, and J. Widmer. Millimeter-wave beam training acceleration through low-complexity hybrid transceivers. *IEEE Transactions on Wireless Communications*, 16(6):3646–3660, 2017.

[14] F. Firyaguna, J. Kibilda, C. Galiotto, and N. Marchetti. Coverage and spectral efficiency of indoor mmwave networks with ceiling-mounted access points. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–7. IEEE, 2017.

[15] R. Ghayoula, N. Fadlallah, A. Gharsallah, and M. Rammal. Phase-only adaptive nulling with neural networks for antenna array synthesis. *IET microwaves, antennas & propagation*, 3(1):154–163, 2009.

[16] S. Gollakota, S. D. Perli, and D. Katabi. Interference alignment and cancellation. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 159–170, 2009.

[17] M. K. Haider, Y. Ghasempour, and E. W. Knightly. Search light: Tracking device mobility using indoor luminaries to adapt 60 ghz beams. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 181–190, 2018.

[18] E. Hamed, H. Rahul, M. A. Abdelghany, and D. Katabi. Real-time distributed mimo systems. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 412–425, 2016.

[19] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. *IEEE transactions on evolutionary computation*, 3(4):287–297, 1999.

[20] H. Hassanieh, O. Abari, M. Rodriguez, M. Abdelghany, D. Katabi, and P. Indyk. Fast millimeter wave beam alignment. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 432–445, 2018.

[21] R. L. Haupt. Adaptive nulling in monopulse antennas. *IEEE transactions on antennas and propagation*, 36(2):202–208, 1988.

[22] R. L. Haupt. Phase-only adaptive nulling with a genetic algorithm. *IEEE Transactions on Antennas and Propagation*, 45(6):1009–1015, 1997.

[23] R. L. Haupt. Adaptive nulling with weight constraints. *Progress In Electromagnetics Research*, 26:23–38, 2010.

[24] J. A. Hejres. Null steering in phased arrays by controlling the positions of selected elements. *IEEE transactions on antennas and propagation*, 52(11):2891–2895, 2004.

[25] H. M. Ibrahim. Null steering by real-weight control-a method of decoupling the weights. *IEEE transactions on antennas and propagation*, 39(11):1648–1650, 1991.

[26] T. Ismail and M. M. Dawoud. Null steering in phased arrays by controlling the element positions. *IEEE Transactions on Antennas and Propagation*, 39(11):1561–1566, 1991.

[27] S. Jog, J. Wang, J. Guan, T. Moon, H. Hassanieh, and R. R. Choudhury. Many-to-many beam alignment in millimeter wave networks. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 783–800, 2019.

[28] S. Jog, J. Wang, H. Hassanieh, and R. R. Choudhury. Enabling dense spatial reuse in mmwave networks. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, pages 18–20. ACM, 2018.

[29] N. Karaboga, K. Güney, and A. Akdagli. Null steering of linear antenna arrays with use of modified touring ant colony optimization algorithm. *International Journal of RF and Microwave Computer-Aided Engineering*, 12(4):375–383, 2002.

[30] S. Karimkashi and A. A. Kishk. Antenna array synthesis using invasive weed optimization: A new optimization technique in electromagnetics. In *2009 IEEE Antennas and Propagation Society International Symposium*, pages 1–4. IEEE, 2009.

[31] M. M. Khodier and C. G. Christodoulou. Linear array geometry synthesis with minimum sidelobe level and null control using particle swarm optimization. *IEEE transactions on antennas and propagation*, 53(8):2674–2679, 2005.

[32] L. Kogan. A minimum gradient algorithm for phased-array null formation. *Radio science*, 40(2), 2005.

[33] J. O. Lacruz, D. Garcia, P. J. Mateo, J. Palacios, and J. Widmer. mm-flex: an open platform for millimeter-wave mobile full-bandwidth experimentation. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 1–13, 2020.

[34] K. C.-J. Lin, S. Gollakota, and D. Katabi. Random access heterogeneous mimo networks. *ACM SIGCOMM Computer Communication Review*, 41(4):146–157, 2011.

[35] C. Lu, Y. Wu, R. Mahmoudi, M. K. Matters-Kammerer, and P. G. Baltus. A mm-wave analog adaptive array with genetic algorithm for interference mitigation. In *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2373–2376. IEEE, 2012.

[36] S. Madani. Nullfi code for null steering. Git Repository, 2021. gitlab.engr.illinois.edu/smadani2/nulling-python.

[37] A. F. Molisch, V. V. Ratnam, S. Han, Z. Li, S. L. H. Nguyen, L. Li, and K. Haneda. Hybrid beamforming for massive mimo: A survey. *IEEE Communications Magazine*, 55(9):134–141, 2017.

[38] T. Moon, J. Gaun, and H. Hassanieh. Online millimeter wave phased array calibration based on channel estimation. *IEEE Design & Test*, 2020.

[39] M. Mouhamadou, P. Vaudon, and M. Rammal. Smart antenna array patterns synthesis: Null steering and multi-user beamforming by phase control. *Progress In Electromagnetics Research*, 60:95–106, 2006.

[40] T. Nitsche, A. B. Flores, E. W. Knightly, and J. Widmer. Steering with eyes closed: mm-wave beam steering without in-band measurement. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2416–2424. IEEE, 2015.

[41] Y. Niu, Y. Li, D. Jin, L. Su, and A. V. Vasilakos. A survey of millimeter wave communications (mmwave) for 5g: opportunities and challenges. *Wireless networks*, 21(8):2657–2676, 2015.

[42] J. Palacios, D. De Donno, and J. Widmer. Lightweight and effective sector beam pattern synthesis with uniform linear antenna arrays. *IEEE Antennas and Wireless Propagation Letters*, 16:605–608, 2016.

[43] L. Poli, L. Manica, P. Rocca, E. Giaccari, and A. Massa. Tolerance analysis with phase errors in linear arrays by means of interval arithmetic. In *EuCAP 2014*. IEEE, 2014.

[44] J. Qiao, L. X. Cai, X. Shen, and J. W. Mark. Stdma-based scheduling algorithm for concurrent transmissions in directional millimeter wave networks. In *2012 IEEE International Conference on Communications (ICC)*, pages 5221–5225. IEEE, 2012.

[45] H. S. Rahul, S. Kumar, and D. Katabi. Jmb: scaling wireless capacity with user demands. *ACM SIGCOMM Computer Communication Review*, 42(4):235–246, 2012.

[46] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez. Millimeter wave mobile communications for 5g cellular: It will work! *IEEE access*, 1:335–349, 2013.

[47] M. E. Rasekh, Z. Marzi, Y. Zhu, U. Madhow, and H. Zheng. Noncoherent mmwave path tracking. In *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*, pages 13–18, 2017.

[48] M. Safe, J. Carballido, I. Ponzoni, and N. Brignole. On stopping criteria for genetic algorithms. In *Brazilian Symposium on Artificial Intelligence*, pages 405–413. Springer, 2004.

[49] R. Schreiber. Implementation of adaptive array algorithms. *IEEE transactions on acoustics, speech, and signal processing*, 34(5):1038–1045, 1986.

[50] R. Shore. Nulling a symmetric pattern location with phase-only weight control. *IEEE Transactions on Antennas and Propagation*, 32(5):530–533, 1984.

[51] G. H. Sim and J. Widmer. Finite horizon opportunistic multicast beamforming. *IEEE Transactions on Wireless Communications*, 16(3):1452–1465, 2016.

[52] S. T. Smith. Optimum phase-only adaptive nulling. *IEEE Transactions on Signal Processing*, 47(7):1835–1843, 1999.

[53] H. Steyskal. Synthesis of antenna patterns with prescribed nulls. *IEEE Transactions on Antennas and Propagation*, 30(2):273–279, 1982.

[54] H. Steyskal. Simple method for pattern nulling by phase perturbation. *IEEE Transactions on Antennas and Propagation*, 31(1):163–166, 1983.

[55] C.-S. Sum, Z. Lan, R. Funada, J. Wang, T. Baykas, M. Rahman, and H. Harada. Virtual time-slot allocation scheme for throughput enhancement in a millimeter-wave multi-gbps wpan system. *IEEE Journal on Selected Areas in Communications*, 27(8):1379–1389, 2009.

[56] S. Sur, X. Zhang, P. Ramanathan, and R. Chandra. Beamspy: enabling robust 60 ghz links under blockage. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 193–206, 2016.

[57] R. Vescovo. Null synthesis by phase control for antenna arrays. *Electronics Letters*, 36(3):198–199, 2000.

[58] T. Vu. Simultaneous nulling in sum and difference patterns by amplitude control. *IEEE transactions on antennas and propagation*, 34(2):214–218, 1986.

[59] T. Wei and X. Zhang. Pose information assisted 60 ghz networks: Towards seamless coverage and mobility support. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 42–55, 2017.

[60] D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[61] Y. Yang, H. S. Ghadikolaei, C. Fischione, M. Petrova, and K. W. Sung. Fast and reliable initial access with random beamforming for mmwave networks. *arXiv preprint arXiv:1812.00819*, 2018.

[62] D. Z. Yong Deng, Yang Liu. An improved genetic algorithm with initial population strategy for symmetric tsp. *Mathematical Problems in Engineering*, pages 1–6, 2015.

[63] J. Zhang, X. Ge, Q. Li, M. Guizani, and Y. Zhang. 5g millimeter-wave antenna array: Design and challenges. *IEEE Wireless communications*, 24(2):106–112, 2016.

[64] R. Zhao, T. Woodford, T. Wei, K. Qian, and X. Zhang. M-cube: a millimeter-wave massive mimo software radio. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[65] A. Zhou, L. Wu, S. Xu, H. Ma, T. Wei, and X. Zhang. Following the shadow: Agile 3-d beam-steering for 60 ghz wireless networks. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2375–2383. IEEE, 2018.

[66] A. Zhou, X. Zhang, and H. Ma. Beam-forecast: Facilitating mobile 60 ghz networks via model-driven beam steering. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

A Appendix

A.1 Pseudocode

Here we present pseudo codes to Nulli-Fi's algorithms discussed in section 4.

Algorithm 1: OPTIMALNULLING$(N, \phi, \phi_0, d, \lambda, \alpha^*)$

$\theta \leftarrow 2\pi \frac{d}{\lambda}\big(cos(\phi) - cos(\phi_0)\big)$;
$v_P \leftarrow \exp(j\frac{N-1}{2}\theta)$;
for n in range(N):
  $v_n \leftarrow \exp(jn\theta)$;
  $\Delta\alpha_n \leftarrow 0$;
for n in range($\frac{N}{2}$):
  if $|\angle(v_P, v_n) - \pi| < \alpha^*$:
    $\Delta\alpha_n = \pi - \angle(v_P, v_n)$;
  $\Delta\alpha_{N-n-1} = \angle(v_P, v_n) - \pi$;
  elseif $\angle(v_P, v_n) < \pi - \alpha^*$:
    $\Delta\alpha_n = \alpha^*$; $\Delta\alpha_{N-n-1} = -\alpha^*$;
  elseif $\angle(v_P, v_n) > \pi + \alpha^*$:
    $\Delta\alpha_n = -\alpha^*$; $\Delta\alpha_{N-n-1} = \alpha^*$;
  if $\angle(v_P, \sum_{n=0}^{N-1} v_n \exp(j\Delta\alpha_n)) \neq 0$:
    return 0;
return $|\sum_{n=0}^{N-1} v_n \exp(j\Delta\alpha_n)|$;

Algorithm 2: CHOOSESUBSET(N, q)

S $\leftarrow \emptyset$;
$(\alpha_0, \cdots, \alpha_{N-1}) \leftarrow$ ideal phase shifts for main lobe;
for n from 0 to $N-1$:
  if α_n is within t degs of an available phase shift:
    add n to S;
return S;

A.2 Phase Calibration

Here we explain Nulli-Fi's phase calibration in detail. In order to calibrate for the difference in the lengths of the wires coming out of each antenna element, we pick one reference antenna element i^*, and calibrate the remaining antennas with respect to this reference. Note that, the process of calibration is finding the additional phase shift one has to apply to antenna j in order to bring it in phase with the reference antenna i^*. To do so, we run a series of simple experiments as follows. We note that throughout the all of these experiments, transmitter and receiver are directly facing each other.

- First, for a fixed i, $0 \leq i \leq 15$, we turn off all antenna elements except i. We then apply phase shifts to the weight of antenna element i over time to cover all the possible phase values, and capture the received power over time. For antenna element i and its n^{th} phase shift $\alpha_i[n]$, we denote the received signal amplitude by $a_i[n]$.
  We repeat this for all i from 0 to 15.
- We repeat the previous phase for all antennas i save for a chosen reference, i^*. Throughout these experiments, we keep element i^* turned on with a constant amplitude a_0, and for the experiment with element i and n^{th} phase shift, we call the corresponding received amplitude $b_i[n]$. Now $b_i[n]$ is the sum of the signals received from i^* and i. If there is a α_{i,i^*} phase shift between the two elements, then we must have

$$b_i(t) = |a_i(t) + e^{\alpha_{i,i^*}} a_0|,$$

  Therefore we can find α_{i,i^*} by performing a simple binary search over all possible values in $[0, 360]$ degrees. An example of the two normalized curves, before and after calibration, is shown in Figure 17.

Algorithm 3: NULLI-FI-GENETIC

Initialize $\mathscr{A} = \{A_1, \cdots, A_M\}$ using
OPTIMALNULLING$(N, \phi, \phi_0, d, \lambda, \alpha^*)$;
if $q \leq 3$:
  $S \leftarrow$ CHOOSESUBSET(N, q);
else:
  $S \leftarrow \{1, 2, \cdots, N\}$;
Limit the adaptive elements to S;
while not converged:
  for i from 1 to M:
    $f_i \leftarrow F(A_i)$;
  sort A_i according to f_i;
  keep $\mathscr{A} = \{A_1, \cdots, A_{\eta M}\}$ and discard others;
 while $|\mathscr{A}| < M$:
    Randomly choose two chromosomes A_i, A_j;
    perform CROSSOVER(A_i, A_j);
    Randomly mutate some A's with prob. p_m;
output A_1.

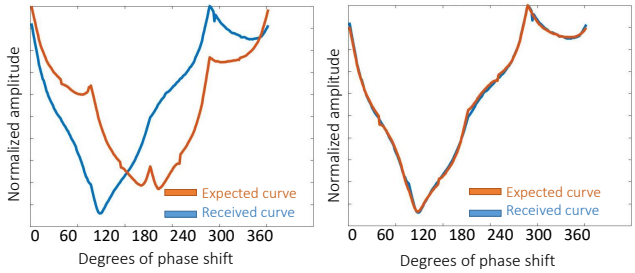


Figure 17: Expected versus measured power of two antenna elements before and after calibration.

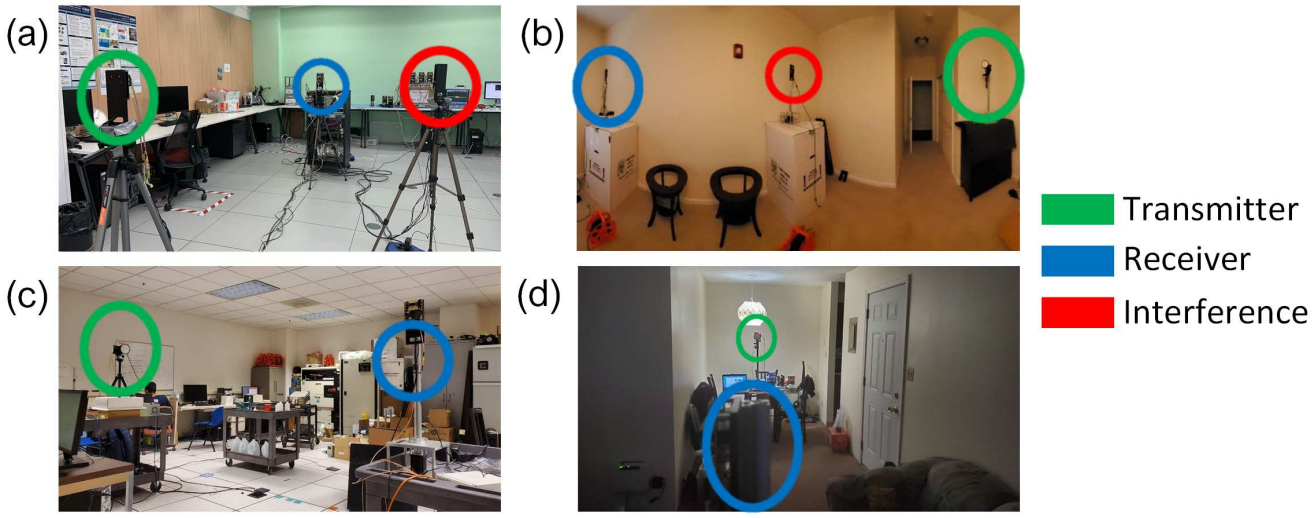


Figure 18: Different locations where we ran our experiments.

A.3 Experimental Setup

Experiment Locations. We ran our experiments in 4 different rooms shown in Fig. 18. Two locations (a, c) were inside a lab environment with many metal cabinets that contributed to multi path reflections. The other 2 locations (b, d) were different rooms inside apartments with many indoor objects. In all rooms, there were human subjects in the background during the experiment, thus constituting dynamic environments.

Nulli-Fi + mm-Flex. One transmitter/receiver pair are implemented using a single FPGA device in a full-duplex manner, i.e. transmitter and receiver functionalities are used at the same time. This pair corresponds to the transmitter and the receiver implementing Nulli-Fi.

We use a second FPGA (mounted on the same hosting chassis) which serves as the interferer. Since both FPGAs are mounted on the same chassis, long cables (5m) are used to carry the baseband signals to the corresponding transmitting antennas (the one transmitting the packets of interest and the one from the interferer). Therefore, with this setup, we are able to easily cover indoor scenarios.

Both FPGAs are managed from a control and management processor integrated in the same hosting chassis. This is used to send/receive frames to/from each baseband processor, configure ADCs/DACs, IP blocks, as well as the setup for the 60GHz Siversima RF-frontends.

A.4 Further Analysis of Nulling Performance

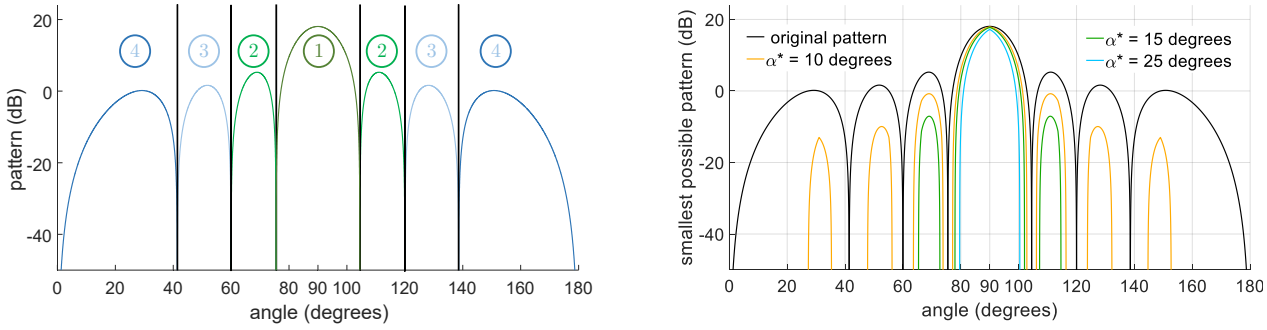
Closed Form Solutions. Alg. 1 offers a step by step solution to find nulls. It is also possible to find closed form solutions for bounds of achievable nulling performance as a function using the algorithm. This can be done by going thorough the algorithm with by keeping the symbol ϕ as opposed to setting it to a specific value. Doing so will result in explicit formulas for the angles for which perfect nulling is possible. For the angles that perfect nulling is not possible, we can find explicit formulas that determine the deepest possible nulls as a Piecewise-defined function of the angle ϕ. Different cases of this piecewise-defined function are separated by the naturally occurring nulls in the original beam pattern. An example of this for $N = 8$ antennas is shown in Fig. 19(a) where there are four cases separated by natural nulls, with each case having its own piecewise formula. For example, Theorems A.1 and A.2 show examples of closed form solutions for nulling around the main lobe as a function of number of elements N, angle of nulling ϕ, the main lobe angle ϕ_0, and the maximum phase shift allowed on each antenna α^*:

Theorem A.1 *The two closest perfect nulls to the main lobe given a maximum phase shift of α^* for each element are given by* $\phi^* = \arccos\left(\cos(\phi_0) \pm \frac{\lambda}{Nd}(1 - \frac{2\alpha^*}{\pi})\right)$.

Theorem A.2 *For the area around the main lobe that perfect nulling is not possible, the deepest possible null at direction ϕ is given by* $N\cos(\frac{N}{4}\theta + \alpha^*)$, *where* $\theta = \frac{2d}{\lambda}(\cos(\phi) - \cos(\phi_0))$.

Specifically, Theorem A.1 determines the areas where perfect nulling is possible, and Theorem A.2 determines the deepest possible nulls for angles where perfect nulling cannot be achieved. For $N = 8$, these formulas correspond to the case 1 in Fig. 19(a). Following similar methods demonstrated in the proofs of these theorems in section A.5 we can find explicit formulas for other cases too.

Using the closed form formulas, we have plotted the best achievable nulling performance (i.e., the lowest possible value of the pattern P for each angle) for $N = 8$ antennas, and $\alpha^* = 10, 15$ and 25 degrees in Fig. 19(b). As revealed by



(a) (b)

Figure 19: (a) The closed form solutions for the single null problem are piecewise-defined functions, with the cases separated by the nulls naturally occurring in the original pattern. Here there are four color-coded cases each corresponding to their own explicit formulas. For instance, Theorems A.1 and A.2 determine the best possible nulling for case 1. (b) Best achievable nulling performance for $N = 8$ elements are depicted for different angles and values of α^*. when a curve is not present at an angle, perfect null (i.e. $P = 0$) is achievable there.

the figure, there is a trade-off between how much we lose in the main lobe, and how strongly we can null different angles. For instance, while $\alpha^* = 15^\circ$ ensures a maximum main lobe loss of 0.3 dB, there are certain regions depicted by the green curve that cannot be nulled. As can be seen, for lower α^* (orange curve) there are more regions that cannot be nulled, while a higher α^* (blue) shows only an area around the main lobe that cannot be nulled. Since Alg. 1 is optimal, we believe it can help decide the degree of trade-off in different applications. This is especially useful as these curves also define as a stopping criterion for our algorithms especially when we lump in the hardware imperfections into the optimization problem. For example if we know that it is not possible to get a nulls stronger than 20 dB in the ideal case (i.e., without hardware imperfections), we can expect that with hardware imperfections the nulling performance cannot get far beyond 20 dB, as we explain in section 4.2.

Alternative Algorithms. In the walk-through example with 6 antennas in section 4.1, the final configuration of the antennas is shown in Fig. 4 (b4). As can be seen from the figure, pairs $\{0,3\}$, $\{1,4\}$ and $\{2,5\}$ are perfectly canceling each other, yielding a null. Looking at this configuration, one might wonder if we can always try and create *pairs* of opposing vectors, such that the sum of every pair is zero. However, it is possible to construct examples where this solution does not work, but Alg. 1 achieves a perfect null. In fact, for larger values of N, it is possible to construct examples in which no set of $K < N$ vectors sum to zero while the sum of all N vectors is still zero. For further information, we refer the interested reader to our git repository where we have implement and compare these algorithms.

A.5 Proofs

Proof of lemma 4.1. We align the main lobe toward some angle ϕ_0, and therefore the signal coming from that angle will sum up coherently. Specifically,

$$|P|^2 = \left|\sum_{n=0}^{N-1} e^{-2\pi j \frac{d}{\lambda} n \cos(\phi_0)} e^{j\alpha_n}\right|^2 = \left|\sum_{n=0}^{N-1} 1\right|^2 = N^2.$$

Imposing additional phase shifts $\Delta\alpha_n$ in order to enable nulling would give us:

$$\begin{aligned}|P'|^2 &= \left|\sum_{n=0}^{N-1} e^{j\Delta\alpha_n}\right|^2 \\ &= \left(\sum_{n=0}^{N-1} \cos(\Delta\alpha_n)\right)^2 + \left(\sum_{n=0}^{N-1} \sin(\Delta\alpha_n)\right)^2 \\ &\geq \left(\sum_{n=0}^{N-1} \cos(\alpha^*)\right)^2 + 0 = N^2 \cos^2(\alpha^*)\end{aligned}$$

since $|\Delta\alpha_n| \leq \alpha^* \leq 90^\circ$. Hence, $|P'|^2 \geq |P|^2 \cos^2(\alpha^*)$ and the loss in the main lobe power is at most $1 - cos^2(\alpha^*) = sin^2(\alpha^*)$.

Proof of lemma 4.2. Replacing θ, we get $v_n = e^{jn\theta}$. Since they are unit vectors, v_k and v_{N-1-k} are symmetric around their sum. Further, we have

$$\begin{aligned}\angle(v_k + v_{N-i-k}) &= \angle(e^{jn\theta} + e^{j(N-1-n)\theta}) \\ &= \angle\left(e^{j\frac{N-1}{2}\theta} \times 2\cos(\frac{N-1-2n}{4})\right) \\ &= \angle e^{j\frac{N-1}{2}}\theta + \angle\cos(\frac{N-1-2n}{4}\theta) \\ &= \frac{N-1}{2}\theta \pm \pi,\end{aligned} \tag{3}$$

where the last line follows from the fact that the phase of a real number is either 0 or π. Since the phase of these vector pair sums are the same (up to $\pm\pi$), so is the sum of all of them, P. This concludes the proof of the lemma.

Proof of Theorem 4.3. For a given nulling angle ϕ, we identify two possible cases. First, if it is not possible to create a perfect null at ϕ, and second, if it is possible to create a perfect null at ϕ.

- In the first case, Alg. 1 does not stop until all vectors $v_0, \cdots, v_{N-1}$ have rotated by $\pm\alpha^*$. In this case, following the exact same argument in the proof of Theorem 4.4, Eq. 6 hold with equality, which means that the best nulling performance is achieved.
- In the second case Alg. 1 where nulling is possible, at some point in the algorithm, $\nu = \angle(v_P, \sum_{n=0}^{N-1} v_n \exp(j\Delta\alpha_n)) \neq 0$ should at some point return π. Otherwise, it remains 0 until the end, in which case nulling should not be possible, contradicting our assumption. Therefore, a some point in the algorithm, $\nu \neq 0$, so the output of the algorithm will be 0, meaning it predicts a perfect null.

In both cases, the output of the algorithm gives the best nulling performance, proving that the Alg. 1 is optimal.

Proof of theorem A.1. For a given ϕ, assume an x-y coordinate for the complex plane, such that $\angle P(\phi) = 0$. In this coordinate, let each vector v_k have the representation (x_k, y_k). We are looking for the first possible ϕ for which there exists a set of additional phase shifts, $\Delta\alpha_k$, such that $P(\phi) = (0,0)$. In its general form, P is expressed as

$$\begin{aligned} P(\phi) &= \sum_n v_n e^{j\Delta\alpha_n} \\ &= \Big(\sum_n \cos((n-\frac{N-1}{2})\theta + \Delta\alpha_n), \sum_n \sin((n-\frac{N-1}{2})\theta + \Delta\alpha_n)\Big) \\ &= \Big(\sum_n x_n, \sum_n y_n\Big), \end{aligned} \tag{4}$$

where θ is defined according to section 4.1. Note that

$$\begin{aligned} x_n^* &:= \min\{\cos((n-\frac{N-1}{2})\theta + \Delta\alpha_n) \mid -\alpha^* \leq \Delta\alpha_n \leq \alpha^*\} \\ &\in \{-1, \cos((n-\frac{N-1}{2})\theta \pm \alpha^*)\}. \end{aligned} \tag{5}$$

Further, we can bound the absolute value of the pattern P as follows.

$$\begin{aligned} |P(\phi)|^2 &= \big(\sum_n x_n\big)^2 + \big(\sum_n y_n\big)^2 \\ &\geq \big(\sum_n x_n^*\big)^2, \end{aligned} \tag{6}$$

where we have bounded the second term with zero. This inequality holds as long as $\sum_n x_n^*$ is positive, which is true around the main lobe, before the first possible null.

Let us rotate each vector v_n to get x_n^* as its x component. Using lemma 1, v_n rotates by $\pm\alpha$ if and only if v_{N-1-n} is rotated by $\mp\alpha$. This means that the two vectors remain symmetrical around the x axis. Therefore, we will necessarily have $\sum_n y_n = 0$, bringing equation 6 to an equality. Hence, as long as $\sum_n x_n^* > 0$, nulling is not possible.

The first point at which nulling becomes possible can therefore be derived by finding the solution to $\sum_n x_n^* = 0$. Using equation 5 combined with lemma 1, we get

$$\sum_{n=0}^{N-1} x_n^* = 2\sum_{n=0}^{\frac{N}{2}-1} \cos((n-\frac{N-1}{2})\theta + \alpha^*) = 0, \tag{7}$$

The solution to which is $\theta = \pm\frac{2}{N}(\pi - \alpha^*)$, or its corresponding ϕ value given in the theorem.

Proof of Theorem A.2. Using Theorem 4.3, we have to run the output of the algorithm for the assumptions in this theorem. Since nulling is not possible, the algorithm will run from 0 to $N-1$, yielding vectors $\cos(n\theta + \alpha^*)$ for $0 \leq n \leq \frac{N}{2}+1$, and $\cos(n\theta - \alpha^*)$ for $\frac{N}{2}+1 \leq n \leq N-1$. Summing them up, we get the result in stated in the theorem.