FREQUENCY SYNTONIZATION FOR DECENTRALIZED DISTRIBUTED PHASED ARRAYS

By

William Reinaldo Torres

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical and Computer Engineering—Master of Science

2025

ABSTRACT

This thesis presents the research and development of an alignment method for spatially diverse, decentralized, distributed phased array systems. The challenge of limited access to reliable alignment signals from a destination or other external primary control system is a significant hurdle for distributed antenna systems in the context of emerging 5G/6G technologies. Even without reference signals, the system is designed to remain fully functional. Decentralized and open-looped wireless sensor networks (WSNs) coordination can mitigate these issues. Still, it necessitates a high level of precision in signal agreement for proper signal operation, such as distributed antenna array beam forming.

This work presents a wireless distributed system intended to complement a decentralized distributed wireless antenna array for communication and remote sensing systems without needing information from second or third-party entities to achieve consensus. The intent is also to demonstrate the partial independence of the syntonization from the synchronization approach. Through the sole use of software in software-defined radios (SDRs), I explore different parameter estimation techniques designed to reduce the residual error when the system achieves frequency consensus, presented as a proof of concept and initial research into such applications that leverage the use of this technique in legacy architecture.

The system uses orthogonal frequency division multiplexing (OFDM) to identify transmitters. It implements an average consensus approach at a system level, which requires minimal prior information from neighboring nodes. This approach ensures that the frequencies converge to the desired residual error tolerance of less than 18°, ensuring constructive interference with a relative power gain of 90% concerning an ideal constructively interfering set of signals.

Copyright by
WILLIAM REINALDO TORRES
2025

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Problem Overview

In recent years, the scientific community has been drawn to imitate biological natural phenomena such as the collective swarm intelligence of ants and birds for technological advances. This has pushed researchers to develop distributed systems of smaller cooperating subsystems that perform better than single platforms and monolithic counterparts [1, 2]. An increase in the need for privacy, security, speed, and resilience on the radio frequency (RF) spectrum has drawn interest from the RF and antenna design engineering community. It has brought interest in developing a spatially distributed antenna system with these capabilities [3–7]. For the system to take full advantage of spatial diversity, signal stability over long-distance separations and prolonged time must be achieved. Hence, the system must actively achieve a signal's wavelength stability without physical cables linking the antennas. This gives rise to a strong interest in achieving signal alignment of distributed systems utilizing RF signals [8].

Departing from classical antenna array theory, which considers a static set of antenna array elements (nodes) in either of the three dimensions (linear, planar, and volumetric) static typical distributions, each with either uniform, non-uniform, random spacing [9], signal coherence is achievable by adjusting each antenna's phase electrical state, to achieve coherence at the moment of transmission or reception.

Extending the single-platform approach to a multi-platform system, each node's frequency [10, 11], phase [12, 13], and timing [14, 15] electrical state must be adequately adjusted to cooperate [7, 16, 17]. In the scenario where the nodes' frequencies are not aligned, a beat pattern is generated, causing regions where the superposition of the signals results in destructive interference (valleys) and other portions of constructive interference. Similarly, when the signals have a phase difference, the summation can result in complete destructive interference. Last but not least, signal or data transmission timing needs to be correctly synchronized to avoid generating a phase difference and resulting in destructive interference, as mentioned previously.

The project detailed in this thesis was conceived with a significant goal: to contribute to designing and developing a system that can stabilize the individual transmitting frequencies from each node of a decentralized distributed phased array system. This system will enable cohesive and constructive interference at any arbitrary destination, significantly advancing RF and antenna engineering by taking advantage of the spectral and spatial diversity of multiple nodes. To achieve the precision and tight requirement presented in [7, 8], it is desired to have signals achieve a maximum phase error of 18°. Distributed antenna systems promise to enhance RF communications [18, 19] and radar [20–25] operations in environments with limited access to reference signals, such as global navigation satellite system (GNSS) [26] or a lack of second-party alignment signals.

1.2 Current Technologies

Current frequency stabilization technologies can be divided into closed and open-loop architectures, each with its consensus method. Furthermore, there are centralized and decentralized architectures, which bring advantages and drawbacks to the syntonization of the network of antenna nodes. Each approach to signal stabilization has limitations that make it less viable for a robust and resilient system, demonstrating the necessity of the presented approach.

1.2.1 Closed Loop Architectures

Closed-loop syntonization requires a third-party beacon that generates a reference signal to which the nodes in the system will syntonize. This reference signal is transmitted to the system, which aligns the electrical parameters as necessary [27–29]. This architecture is a closed-looped centralized architecture because it depends on the third-party reference signal or signal information to control the system. The alignment information can be transmitted to the system in two ways: centralized and decentralized.

The centralized approach uses a leader node in the system to relay the information to align with the other nodes. Thus, the beacon system only needs to constantly communicate with the leader, and the leader with the followers. This method leaves a single point of failure, jeopardizing the system. On the other hand, the decentralized approach requires communication between the beacon and each node of the phased array. One beacon and multiple nodes mean that a schedule is

necessary for the communication among them, requiring high timing precision but also introducing significant latency to the operation, which can affect the oscillator drifts [30–32].

The drawback of a closed-loop architecture is its inherent limitation on beam steering. Since the nodes calibrate their parameters around the beacon, they must recalibrate themselves when steering to a different destination, and this requires prior knowledge of the location of the new destination, knowledge that is not always available to the system. Then, the system must steer the antenna's beam, limiting the direction in which the distributed system can focus the beam. This enables the signals to constructively interfere in the beacon's direction and nowhere else, limiting the applications to wireless communications. Using Fig1.1 as an example, the four nodes in the system can only align their parameters with respect to the destination, and nowhere else, because they highly depend on it.

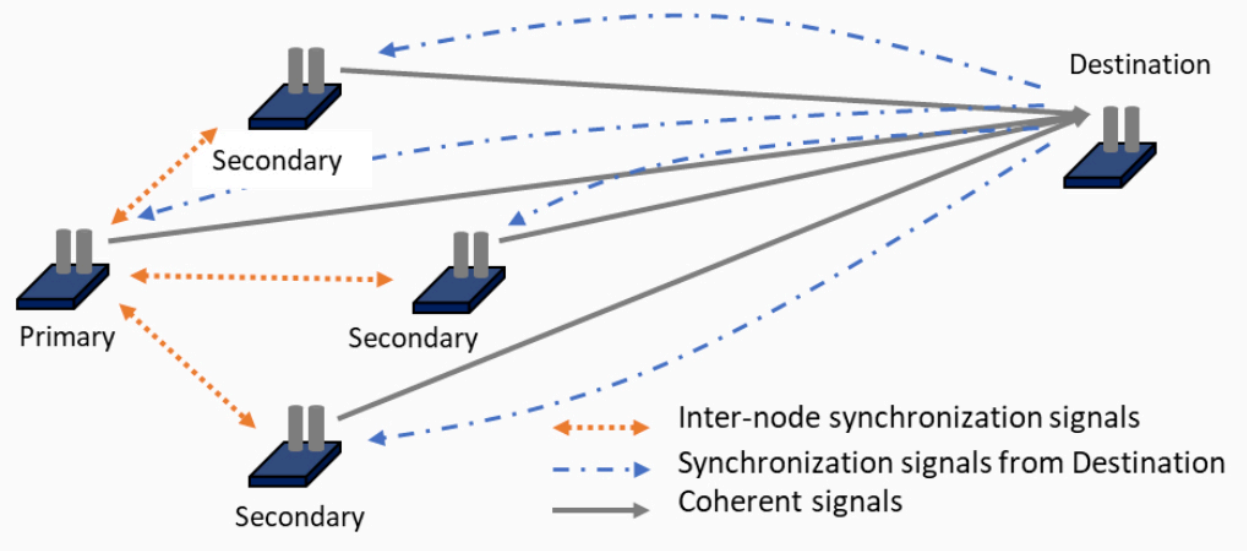

Figure 1.1 Four Node Closed Loop System [33]

1.2.2 Open Loop Architectures

As the name suggests, open-loop architecture does not rely on a second or third-party beacon signal for the system to agree upon its consensus values; the calibration of the signal's electrical parameters happens locally among the different nodes that make up the system. The electrical parameters are aligned locally at the system level. Because they do not depend on external signals, modifying the parameters for beam steering, forming, or nulling at arbitrary locations is trivial. With the ability to perform arbitrary beam-forming, radar and communications applications are possible.

Also, reducing the dependency on a beacon synchronization signal, a single point of failure, and scheduling the communications links between the beacon and nodes that make up the system helps make the system more resilient in harsh environments. To put it in perspective, Fig1.2 shows a four-node system where the nodes align themselves without receiving any information from a third-party beacon. This allows the system to beam steer, enabling more capabilities.

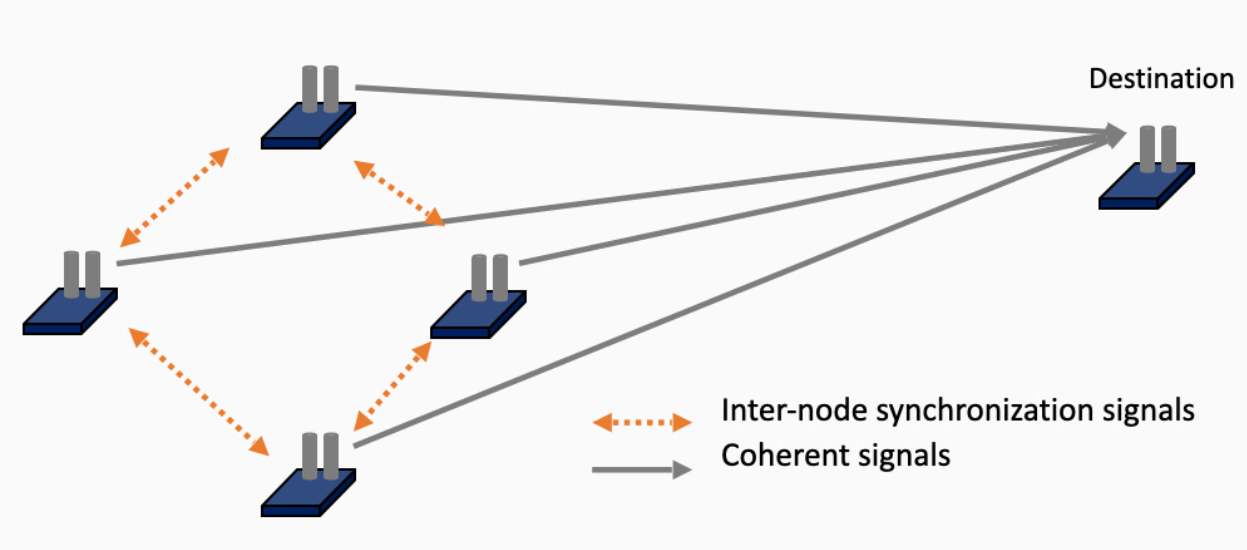

Figure 1.2 Four Node Open Loop System [33]

1.2.3 Centralized Architecture

The leader-follower dynamic is another topology for distributed phased arrays, as foreshadowed in 1.2.1. It is also called centralized architecture because it makes up a central node, a leader, and the other nodes or subnetworks are followers. In this network setup, the leader, as part of the system's nodes, relays directly the values of the electrical parameters to the followers and ensures they are correctly aligned. The system leader does the most significant portion of the work because it needs to act as a relay between the destination and the other nodes, or if no feedback is provided, it needs to determine and align the signals by itself, while the followers are just complying. A significant drawback of centralized architecture is the fragility of the single point of failure, which is the leader node failing. In the case of a targeted attack or downtime of the leader node, the entire system will be vulnerable and not operational. This would defeat the whole purpose of distributing the phased array.

1.2.4 Decentralized Architectures

The decentralized architecture does not rely on a primary node that leads the other nodes with the parameters alignment; the alignment occurs concurrently among the nodes, and they attempt to align to an expected value. A decentralized network addresses the issue of a single point of failure brought up in Sec1.2.3 because no single node is more important than the others; therefore, the system can continue operating when one or several nodes have failed or are down due to maintenance, without having a significant impact in the performance of the system.

A downside of a decentralized system is the limited information sharing among nodes. Addressing the problem of limited information propagation by enforcing tighter constraints in the network [34–38] (explained in more detail in Sec2.2) demonstrates the resilience to link or node failure because no link or node is imperative.

1.3 Contribution of this Work

The work presented in this thesis focuses on contributing to the development of a physical demonstration of a distributed, decentralized, open-looped phased array system that is resilient to single point failure, as mentioned in Sec1.2.4, and with both communication and radar capa-

bilities, as addressed in Sec1.2.2. This project aims to achieve inter-node signal syntonization of two-tone waveforms with a residual error of less than 18° [7, 38–40] purely through software, utilizing software-defined radios, which should allow for distributed beamforming in the future. It also aims to demonstrate that it is possible to maintain consistently updated accuracy, as mentioned earlier, with frequency orthogonal signals while frequency hopping to aggregate another level of security from man-in-the-middle and interference attacks [41–45].

CHAPTER 2

THEORY AND SIMULATION METHODS

2.1 Introduction

Phased arrays are composed of multiple antennas operating coherently with transmission and reception capability. When the antennas that compose the array are placed at a significant distance from each other, it is not convenient to have them synchronized with extremely long cables, which can limit the applications where the system can be used. This is when the interest in a distributed system starts. Because the system is spatially distributed, each antenna has its power source and local oscillator and must be independent of its neighbors but still contribute to the system. To be able to achieve coherence, the signals must have a difference between their phases of less than $18°$, or $\frac{\pi}{10}\ rad$, to demonstrate the capabilities of stabilization of two-tone signals for real-time applications, and without time alignment using a decentralized average consensus approach.

2.2 Network Model

The antennas in the system are spatially separated; hence, the system can be described using a graph, $G = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of vertices corresponding to the antennas node, and $\mathcal{E}$ represents the set of edges connecting the vertices, and the communication links and line of sight of the transmitting antennas, and $\mathbf{A}$ is the adjacency matrix [46]. The graph used to describe the system network can be directed [47] or undirected [39], greatly impacting the necessary information and the node's information from its neighbors. The graph that describes the network must be strongly connected, which means that the information can flow from an origin node to any other node in the system through intermediate nodes without getting stuck. The adjacency matrix, where the columns correspond to the nodes that are transmitting, and the rows the receiving nodes, that best describe these requirements are $1)$ symmetrical over its diagonal, which takes care of the undirected communication and robust connectivity $\mathbf{A} = \mathbf{A}^T$. Another condition that the connectivity matrix needs to comply with is $2)$ self-loop, with non-zero elements along the

matrix's diagonal. Also, the network must be $3)$ decentralized, allowing for cells to be populated with zeros for nonexistent links between nodes. As shown in table2.1, an adjacency matrix, **A**, describes the link connections between the nodes. Additionally, the adjacency matrix is adjusted to

RX \TX Node	0	1	2	3	4	5	6
0	1	1	0	0	1	0	1
1	1	1	0	0	1	0	1
2	0	0	1	0	1	0	0
3	0	0	0	1	1	0	0
4	1	1	1	1	1	0	1
5	0	0	0	0	0	1	1
6	1	1	0	0	1	1	1

Table 2.1 A sample strongly connected seven node adjacency matrix for fig 2.1

assign weights to the corresponding channel's data and to compute the node's local weighted arithmetic mean. The only constrain for assigning the weights to the adjacency matrix and converting to a mixing matrix, **W**, is that the matrix needs to be double stochastic when the network communication links are undirected [39], and row stochastic when the network has directed links [47]. Focusing on an approach that enables local arithmetic averages with the neighboring nodes, we use the Metropolis-Hasting algorithm [48] to generate the mixing matrix. In the interest of this project, to test the scalability of orthogonal frequency signal syntonization, we assume simultaneous transmission, quasi-static nodes, and bidirectional communications that comply with all the requirements for the system mentioned previously.

$$W_{i,j} = \begin{cases} \frac{1}{max\{deg(i),deg(j)\}+1}, & if\ (i,j) \in \mathcal{E}, \\ 0, & if\ (i,j) \notin \mathcal{E}\ and\ i \neq j, \\ 1 - \sum_{j:j\neq i} w_{i,j}, & if\ i = j \end{cases} \tag{2.1}$$

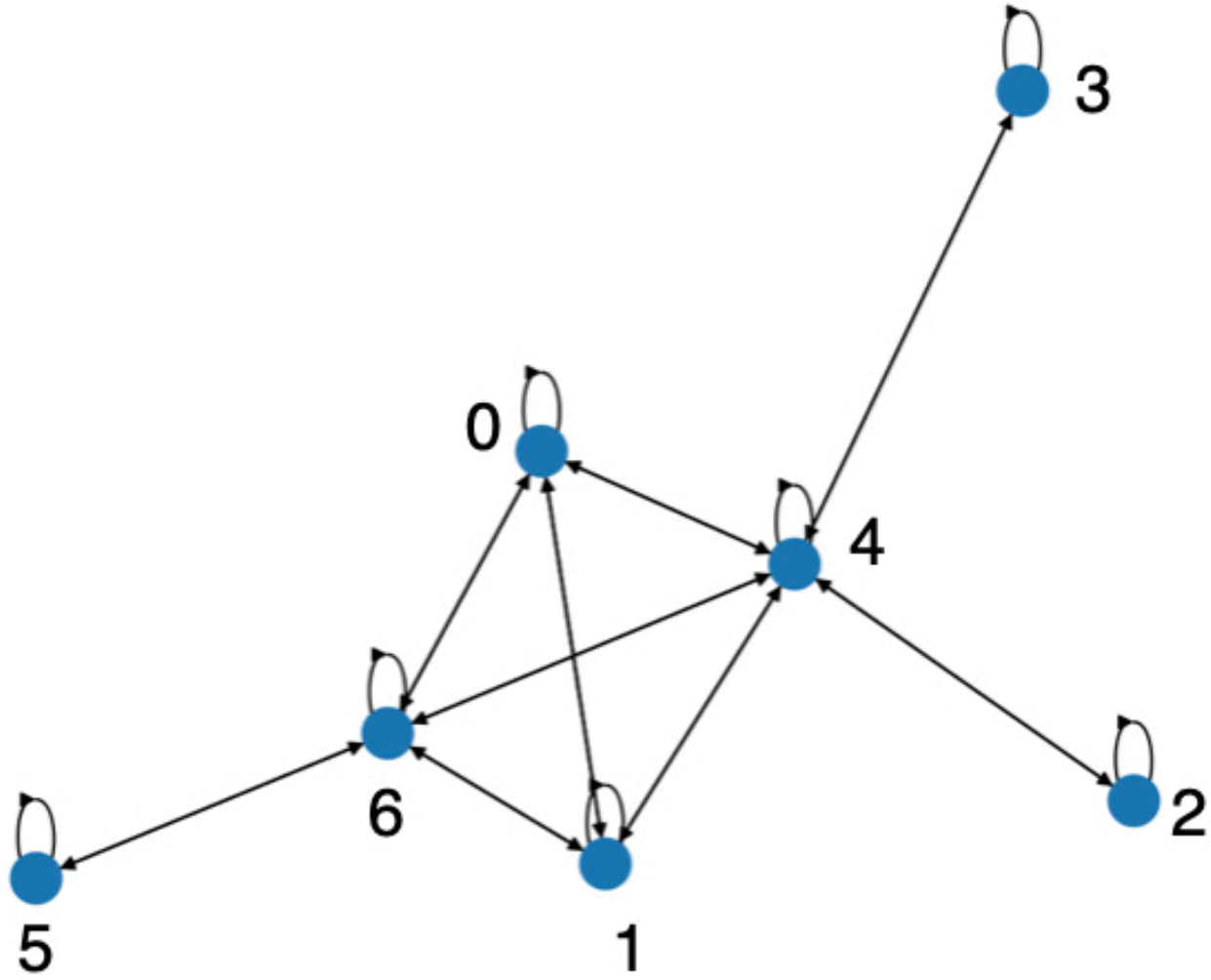

Figure 2.1 Strongly connected network with seven independent nodes with undirected/bidirectional communication links among themselves.

RX \TX Node	0	1	2	3	4	5	6
0	$\frac{23}{60}$	$\frac{1}{4}$	0	0	$\frac{1}{6}$	0	$\frac{1}{5}$
1	$\frac{1}{4}$	$\frac{23}{60}$	0	0	$\frac{1}{6}$	0	$\frac{1}{5}$
2	0	0	$\frac{5}{6}$	0	$\frac{1}{6}$	0	0
3	0	0	0	$\frac{5}{6}$	$\frac{1}{6}$	0	0
4	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	0	$\frac{1}{6}$
5	0	0	0	0	0	$\frac{4}{5}$	$\frac{1}{5}$
6	$\frac{1}{5}$	$\frac{1}{5}$	0	0	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{7}{30}$

Table 2.2 Sample mixing matrix, $\mathbf{W}$, for fig 2.1

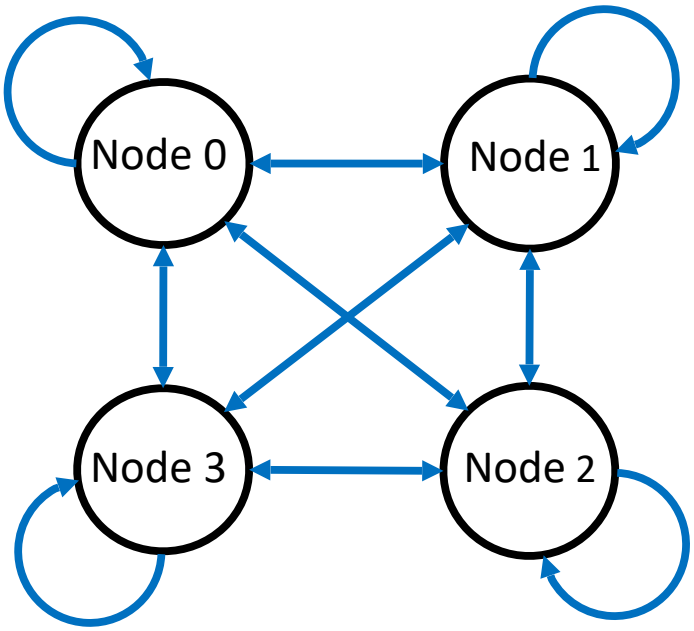

Figure 2.2 Fully connected, looped, network composed of four nodes with self-talk

2.3 Signal Model

As the name suggests, two-tone waveforms, described by equation 2.2, are composed of the superposition of two single-tone signals at baseband with different frequencies, f_1 and f_2.

$$S(t) = A_1(e^{j\cdot 2\pi\cdot f_1\cdot t}) + A_2(e^{j\cdot 2\pi\cdot f_2\cdot t}) \tag{2.2}$$

This waveform is modulated onto a carrier signal. The advantages of using two-tone waveforms have been extensively analyzed for electronic systems calibration techniques, simultaneous radar and communications applications [49–51] with tracking capabilities down to $15\,\mathrm{mm}$. Two-tone waveforms have been used to syntonize nodes by modulating a single frequency of $10\,\mathrm{MHz}$ into the signal's bandwidth, and transmitting it to neighboring nodes with a self-mixing circuit which takes advantage of products of sinusoids to demodulate the single frequency [52, 53]. The bandwidth is the magnitude of the difference between the tones that compose the waveform, like in equation 2.3

$$\Delta f = |f_2 - f_1| \tag{2.3}$$

The downfall of these frequency-sharing approaches is that they require additional hardware for proper demodulation of the reference frequency from the bandwidth, which significantly increases the cost, limiting the speed of scaling to an extensive system, or an already deployed system, and requiring scheduling the times each internode syntonization signal can be transmitted, affecting the purity of the signal generated by the oscillators [30–32]. Contrary to the earlier work, this project focuses on achieving a similar feat without requiring external hardware and purely on the digital signal processing capabilities of off-the-shelf hardware such as software-defined radios.

Two essential characteristics of the two-tone waveform are the difference between the two frequencies, the bandwidth, and the midpoint of the frequency values. Traditionally, the two frequency values have been equally distant from $0\,\mathrm{Hz}$ as double sideband signals, but in the interest of this project, the signal remains double sideband but not centered at the carrier; instead, they will be centered arbitrarily, away from the carrier, in different channels of the transceivers instantaneous bandwidths allowing simultaneous reception and transmission.

$$\frac{f_1 + f_2}{2} \neq 0 \tag{2.4}$$

2.3.1 Transmit Signal Model

Each node is scheduled to transmit simultaneously but spread across the spectrum in its predetermined channel, obeying orthogonality conditions, leaving a band guard among tones, as depicted in figure 2.3. The signals are assumed to be planar waves that can be modeled using complex exponentials. As proof of concept and validation in simulation, it is believed that the internode syntonization signals were initially transmitting pulses outside the detectable portion of the spectrum that the nodes could scan; hence, there is an initial error in the waveform's tones and consequently in the bandwidth. Then, they all agree to syntonize in some predetermined portion of the spectrum, such that their signals occupy the instantaneous bandwidth the nodes can sample. For instance, an over-the-shelf software-defined radio, such as National Instruments (NI) Ettus X310, with UBX160 daughter boards that can sample at $200\,\mathrm{MSa/s}$, hence the transmitters divide the portion of the spectrum among themselves leaving an approximately equally spaced

channel where the signals will coexist. To obey the Nyquist criterion [54], the signals can only exist between the carrier frequency and $100\,\mathrm{MHz}$ away from the carrier frequency. After the sub-channels are allocated, the two-tone central transmission value is determined at the middle of the channel, separating the signal's tones enough for orthogonality. Thus, equation 2.2, which describes a continuous two-tone waveform, is rewritten to equation 2.5, where f_c corresponds to the channelization offset which centers the two-tone's frequencies away from the carrier frequency and the other inter-node aligning signals and allows identification if frequencies hopping is necessary [42–45, 55–57]. Also, Δf corresponds to the two-tone waveform bandwidth as defined in equation 2.3, where f_1 is considered the higher frequency tone and f_2 the lower frequency tone, and A_1 and A_2 are their corresponding single tone amplitudes.

$$S(t) = A_1\, e^{j2\pi(f_c+\frac{\Delta f}{2})t} + A_2\, e^{j2\pi(f_c-\frac{\Delta f}{2})t} \tag{2.5}$$

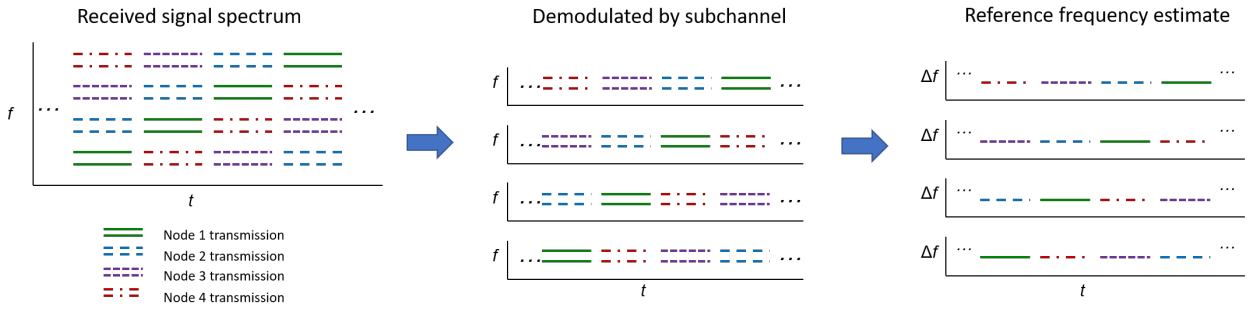

Figure 2.3 A graphical representation of a transmission schedule for two-tone signals within the same portion of the electromagnetic spectrum, and its reference frequency demodulation

As part of a larger project working towards the electrical parameter alignment of distributed antenna array signals, it is necessary first to demonstrate syntonization independent of signal synchronization and phase alignment, but not completely excluding them from the equation. This is why, to work along with a similar system designed in [58–60], the signals will be pulsed for the security of internode aligning signals. This means that to go around the problem of time misalignment among nodes and reduce the impact it may have on signal syntonization's proof of concept, the two-tone waveform is pulsed, and equation 2.5 is modulated onto a rectangular wave of duration t. Because each node believes they are synchronized to have at least one

aligning iteration, the transmitting is appended to the leading and trailing zero pad to position the signals in the middle of the transmit window, as described in equation 2.6.

$$S[n] = \begin{cases} 0 & \text{if } 0 \leq n \leq z \\ A_1\, e^{j2\pi \frac{f_c + \frac{\Delta f}{2}}{f_s} n} + A_2\, e^{j2\pi \frac{f_c - \frac{\Delta f}{2}}{f_s} n} & \text{if } z < n \leq z + l \\ 0 & \text{if } N - z < n \leq N \end{cases} \tag{2.6}$$

Equation 2.6 is the ideal discrete single-channel transmit signal, where f_s is the sampling frequency at the transmitter, N is the length of the transmit window, l is the length of the signal pulse, and z is the total length of the both leading and trailing zero padding divided by two. To reiterate, and for clarity, the reason for zero padding is to mitigate the impact of time misalignment of the system, which is not the focus of the research. At a node level, the node passes equation 2.6 through a digital-to-analog converter (DAC) and modulates that signal to a carrier signal.

2.3.2 Receive Signal Model

Each node receives its neighbor's signal that has passed through a noisy channel. Ideally, highly directive antennas would be used, but this would limit the speed and cost at which the system can scale. Hence, the use and simulation of omnidirectional antennas are implemented to achieve a similar result to a real-world scenario. This means that because transmission and reception happen at the same internal clock tick of each node, the node will also receive its signal from self-interference. In the scenario of a two-node system, equation 2.7 corresponds to the single communication channel between two nodes without self-interference. In contrast, equation 2.8 encompasses the superposition of all the neighbor's internode syntonization signals, including the node's local signal, where j is the identification index for transmitting nodes, and i is the identification index for receiving nodes.

$$S_j[n] = \begin{cases} w_{i,j}[n] & \text{if } 0 \leq n \leq z \\ B_1\, e^{j\cdot 2\pi \cdot \frac{f_c + \frac{\Delta f}{2}}{f_s} \cdot n} + B_2\, e^{j\cdot 2\pi \cdot \frac{f_c - \frac{\Delta f}{2}}{f_s} \cdot n} + w_{i,j}[n] & \text{if } z < n \leq z + l \\ w_{i,j}[n] & \text{if } N - z < n \leq N \end{cases} \tag{2.7}$$

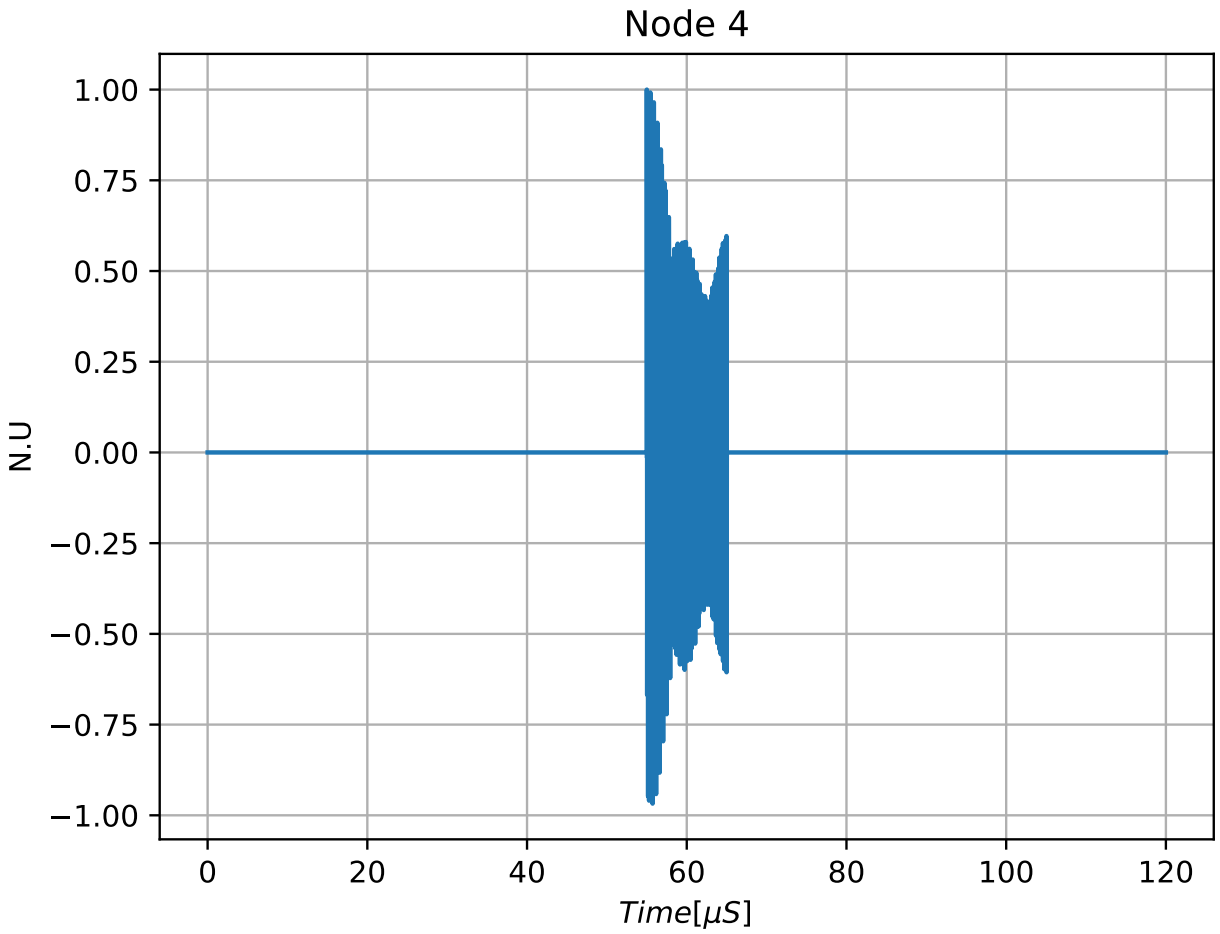

Figure 2.4 A sample simulation, in the time domain of the pulsed digital signal, a single node will send through the DAC and then modulate to the carrier signal.

For equation 2.7, $w_{i,j}[n]$ is assumed to be zero-mean additive white Gaussian noise inherited in the channel the signal travels where i is the transmitting node, and j is the receiving node. It is essential to highlight that although the nodes receive the same signals, each signal is traveling through its own channel in the direction of their neighboring nodes; hence, $w[n]$ in equation 2.7 is different for each transmitted signal and their corresponding receiving node, this is contrary to the approach proposed by [39, 47] where they assumed that each node detected the same channel noise, and interference among alignment signal because their approach depends on time division multiplexing access.

$$S_{i,RX}[n] = \sum_{j=1}^{J} S_j[n] \tag{2.8}$$

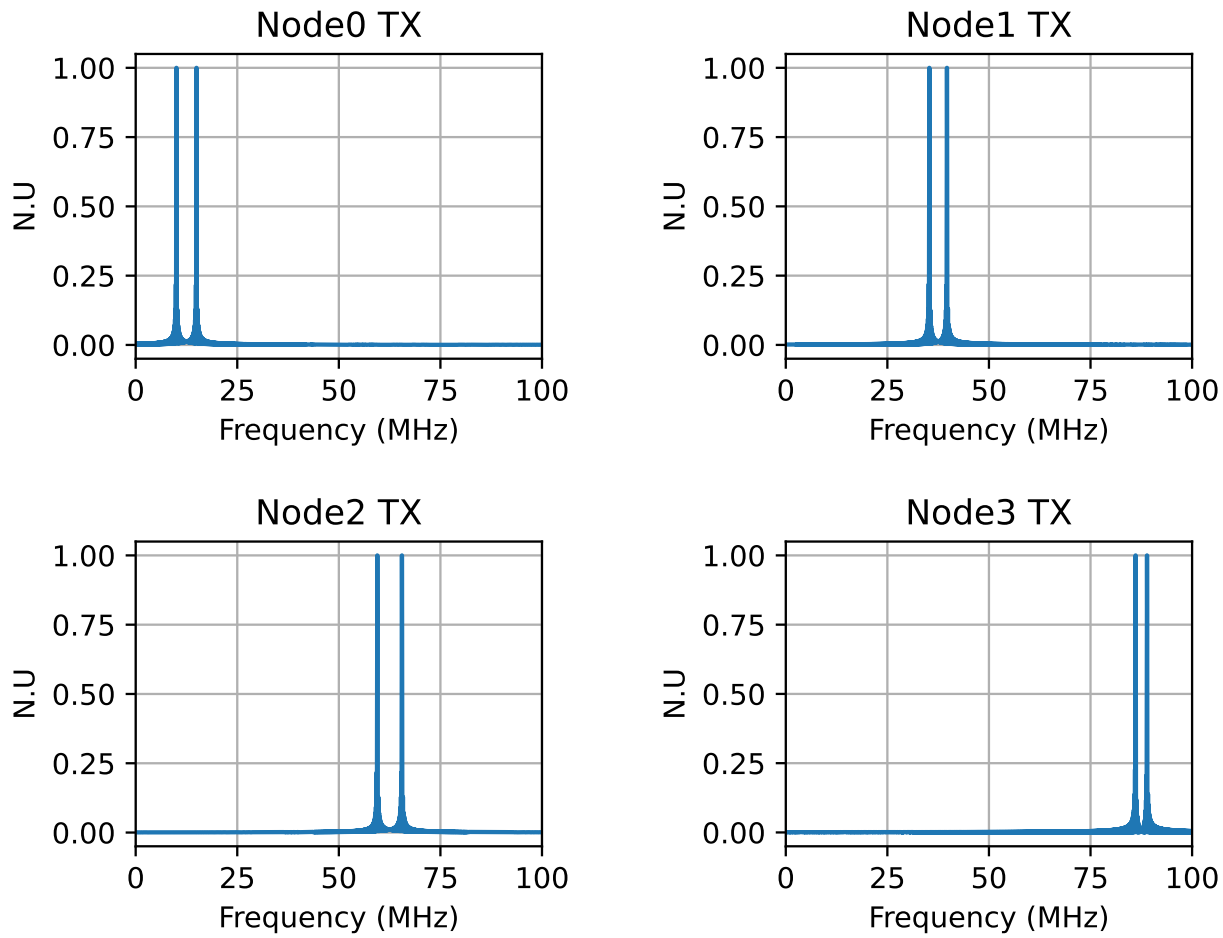

Figure 2.5 A sample simulation, in the frequency domain of the pulsed digital signal, a single node will send through the DAC and then modulate to the carrier signal.

2.4 Parameter Estimation

Now that the nodes have received the signals, it is necessary to evaluate the electrical parameters of the tones individually and collectively concerning the two-tone waveform to which they belong. As part of transitioning distributed phased array alignment to solely digital signal processing, this work focuses on bandwidth syntonization; hence, it is crucial to look for single-tone frequency estimation techniques and how they can be extended to real-time multi-tone estimation techniques.

2.4.1 Cramer Rao Lower Bound

The Cramer-Rao lower bound (CRLB) is the variance of any unbiased estimator for a parameter in a statistical model. Although many bounds exist, such as the Barankin bound [61,62],

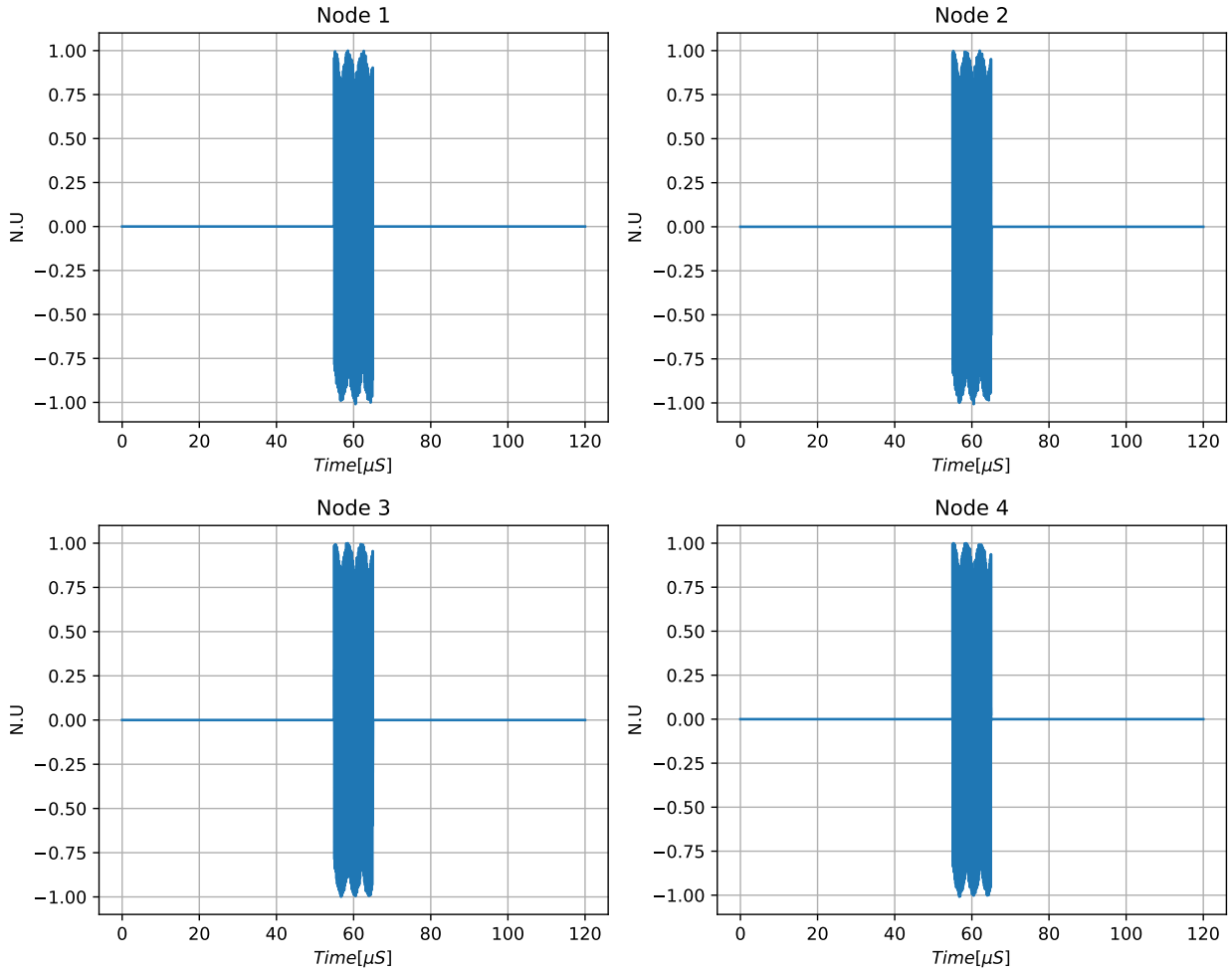

Figure 2.6 A sample capture, in the time domain of the pulsed digital signal, each node will send through the DAC and then modulate to the carrier signal.

in theory, it is the most obvious to determine if an estimator exists capable of attaining the bound. In layman's terms, the CRLB is the lower bound that an estimator can achieve. In the interest of this project, it is necessary to assume that the single signal is deterministic and is submerged in additive white Gaussian (AWG) noise. The derivation of the CRLB for a single sinusoidal signal is well known and has been derived multiple times in the literature by [63, 64], and they determined the CRLB, inequality 2.10, from equation 2.9.

$$x[n] = A\,cos(2\pi f_0 n + \phi) + w[n] \qquad n = 0, 1, \ldots, N-1 \tag{2.9}$$

$$\text{var}\left(\hat{f}_0\right) \geq \frac{12 \cdot f_s^2}{(2\pi)^2 N^3 SNR} \tag{2.10}$$

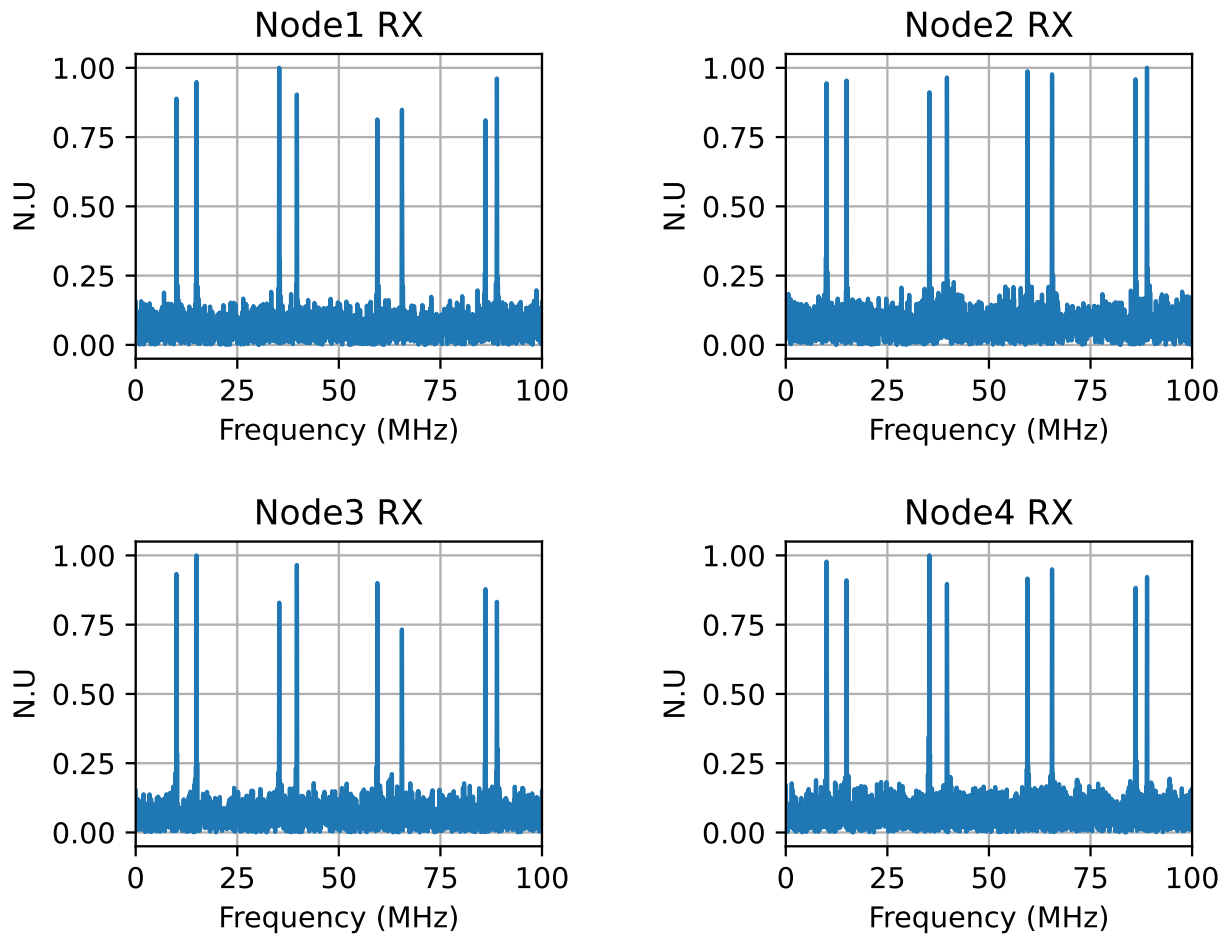

Figure 2.7 A sample capture, in the frequency domain of the pulsed digital signal, a single node will send through the DAC and then modulate to the carrier signal.

In [65], Richards derives the CRLB assuming that the signal is a complex exponential in AWG noise from equation 2.11 and ends up with the variance, inequality 2.12, half of that derived by Van Tree and Kay. For both, equation 2.9, 2.11, f_0 is the normalized frequency and is equivalent to $\frac{f}{f_s}$, where f is the signal's frequency and f_s is the sampling frequency.

$$x[n] = A(e^{j\cdot 2\pi f_0 n+\phi}) + w[n] \qquad n = 0, 1, \ldots, N-1 \tag{2.11}$$

$$\text{var}\left(\hat{f}_0\right) \geq \frac{6 \cdot f_s^2}{(2\pi)^2 N^3 SNR} \tag{2.12}$$

For equations 2.10 and 2.11, the CRLB is trivially obtained because the estimator of a single sinusoid is unbiased but with the existence of multiple sinusoids obtaining a close form expression

for the CRLB or Barankin bound is mathematically not feasable, and a further analysis will be done on the phenomena of spectral leakage in section 2.4.3, which gives rise to the biased.

2.4.2 Methods for single tone estimation

The interest in parameter estimation has developed into a full area of research, and several single approaches for parameter estimation have been developed. The most popular reference algorithm researchers look into is the methods derived by Steven Kay [66], which are done in the time domain. Contrary to Kay's method of estimating the frequency in the time domain, [67] approaches the problem with an initial coarse frequency estimate and refinement using calculus and geometry to interpolate the discrete Fourier domain and calculate the signal's frequency. Similarly to [67], [68] depends on the coarse frequency estimate and further refinement using the secant method. [5, 69, 70] also achieve or closely approach the CRLB at high SNR when estimating a single sinusoidal frequency. It is important to reiterate that these methods rely heavily on the geometry of the sinusoid and its inherited characteristics to achieve the CRLB, given that a single sinusoid estimator is unbiased, as explained in section 2.4.1.

2.4.3 Parameter Estimation Extension

In the presence of multiple sinusoids with a finite duration and an unknown future update value that may not lie at a frequency bin, mutual interference between the tones is inevitable. This interference is known as a frequency or spectral leakage. As depicted in Fig.2.8, the a's correspond to the time domain transmit signal, while the b's represent the Fourier domain. The summation of two monotone signals generates a two-tone waveform. Figure 2.9 presents the superposition in the Fourier domain of both single tones and their corresponding combination. Although it seems that the plots align properly, there is an impact on the plotted sinc function, affecting the peak's location and, hence, any geometric estimate in the Fourier domain. The distortion becomes more prominent with an increase in the number of frequencies present during the signal capture. The estimation of multiple frequencies introduces bias to every tone; hence, a single estimator won't be able to achieve the CRLB directly. That is why isolating the tones from each other is necessary to utilize the single-frequency estimation techniques. This means the signals must be passed

through filters to isolate the tones. No external hardware should be essential for this project; hence, narrowbandpass finite impulse response (FIR) filters are considered. As demonstrated in [71], to approximate the CRLB with estimators, it is necessary to make a two-pass filtering approach, where the sole purpose of the first pass is to obtain a coarse estimate of the frequency to locate the center of the filter. Once the filter is designed, and the signal is passed through the filter, another coarse frequency estimate, followed by a refinement or interpolation approach, yields an estimate that approaches the CRLB. This two-way approach is implemented in every single tone. Thus, the time complexity of frequency estimation grows with the number of tones present. Another way to estimate the frequencies is to use super-resolution estimation techniques, such as the Pisarenko method [72], ESPRIT [73], and MUSIC [74]. The literature has analyzed all of these techniques, and the consensus is that even though they significantly approach or achieve the CRLB estimation error, the computation cost is too high because they all require Eigenvalue decompositions and large matrix inversions to obtain the frequency estimate. Also, the nodes must know a priori the quantity of frequencies they are estimating, which increases the computation time to determine the value initially and limits the number of applications where it can be implemented. Because a single computer controls the nodes for the coarse time alignment, it also means that the computer does all the necessary processing, meaning that the time complexity of any estimator is scaled by a factor of n, where n is the number of nodes that compose the system.

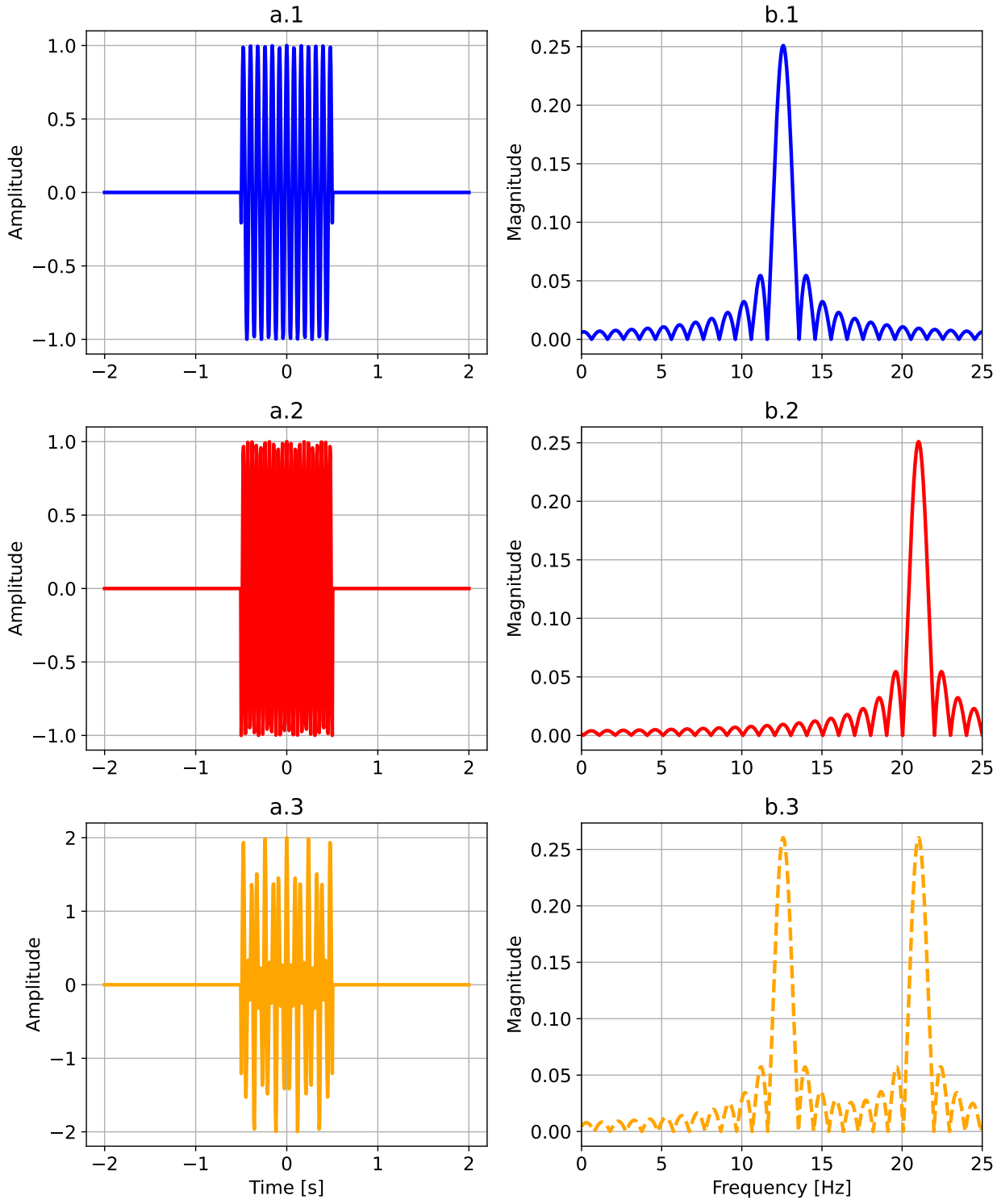

Figure 2.8 A simulation of the superposition of two single-tone waveforms to create a two-tone waveform, both in time and frequency domain

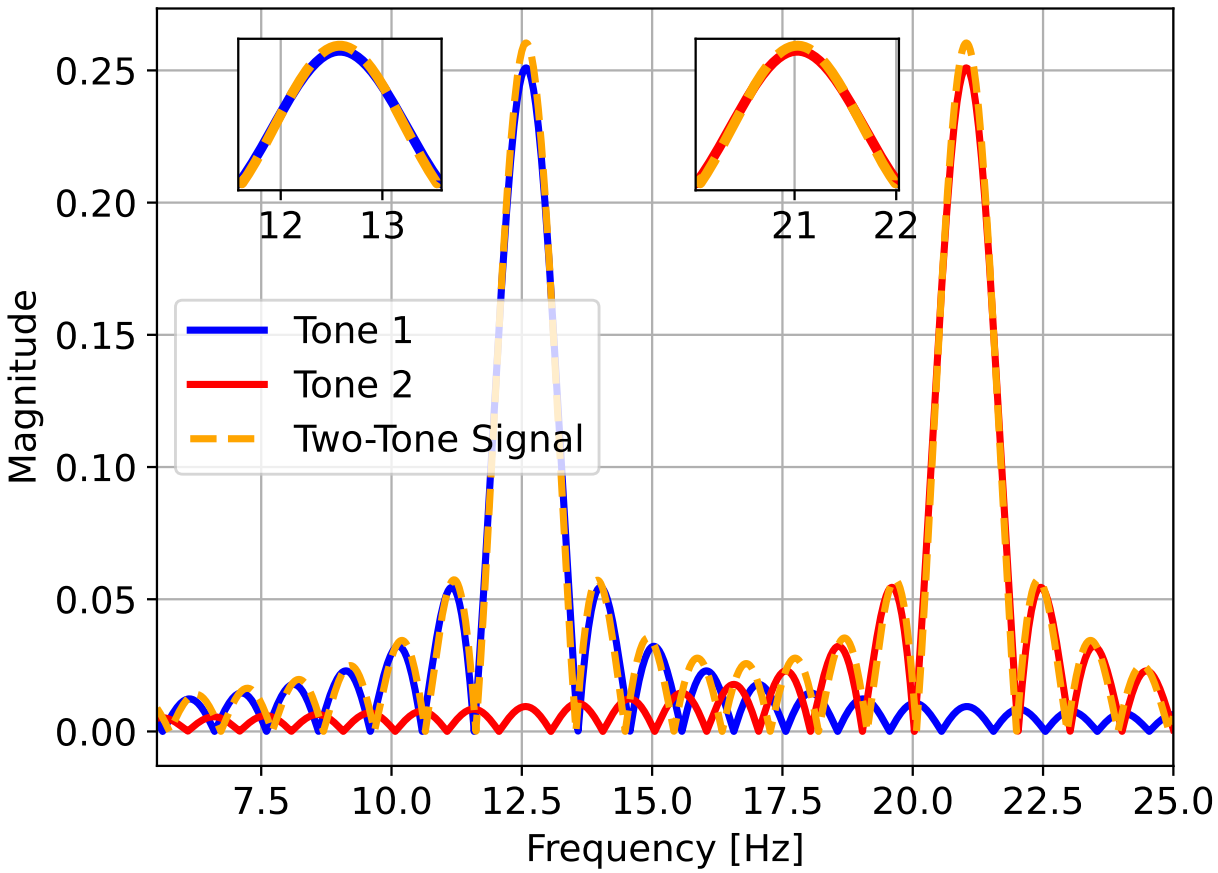

Figure 2.9 A simulation comparison of two single-tone waveforms to a two-tone waveform

2.5 Average Consensus Algorithm Simulation

Once the nodes have calculated the received signal's bandwidth and identified the transmitting node, they can calculate the weighted arithmetic average of the bandwidths and their own transmit bandwidth. A Python3 script was programmed for the system simulation, APPENDIX I. Now that the network has been designed, the nodes must generate the signals they transmit. The code assumes that all nodes transmit and receive simultaneously and that signal propagation is negligible. Although this assumption is not valid during the physical experiment, it greatly facilitates the processing of the signals in both the simulation and the physical experiment. The simulation also assumes that the nodes are identical, have the same transmit and receive duration, which are not the same for both transmit and receive, and have the same sampling rate. For a realistic approach, by default, the nodes transmit at the same carrier frequency and have a sampling rate

of $200\,\mathrm{MSa/Sec}$; hence, the Nyquist criterion determines that the highest frequency component cannot exceed $100\,\mathrm{MHz}$. With the Nyquist rate in mind and leaving enough spectral space away from the DC noise, the spectrum is divided equally among the number of nodes to determine the channel at which they will transmit. After deciding which band of the spectrum they will be allowed to transmit on, the midpoint of the band becomes the channelization offset, f_c, as explained in Section 2.3.1 equation 2.5, and section2.3.2 equation 2.7 and equation 2.8. After determining the central frequency value, the transmission channel's bandwidth, and the band guard bandwidth for the worst-case scenario the bandwidth, the simulation generates random values for the initial bandwidths from a random normal distribution with limits between half the channel's bandwidth, the highest frequency respecting the band guard, and 10% of the bandwidth. Then, the bandwidth and central frequency values are assigned to their corresponding transmitter, and the two-tone time complex sinusoid is generated in time domain signals for the designated pulse duration, which should be less than or equal to the receiver capture window duration.

Once the signals have their independent noise, they are added to each other as explained in equation 2.8. Now that the node has received the superposition of the signals, it will strip away any zero padding because it contains channel noise, which will impact the estimation more than desired. In the simulation, a threshold value cuts off the noise. Hence, the minimum SNR considered viable for the experiment must be greater than $0\,\mathrm{dB}$. Now that just the superpositions of the signals are left, the estimation algorithm is implemented. As addressed in Section 2.4.2, any geometric method to approximate the frequency estimation should be good enough [5, 67–69] for the simulation results presented in Section 2.6, [5]'s quadrature least square (QLS) approach for peak finding was chosen as the most efficient approach compared to the others. The advantage of this technique is that it is a very similar estimation error to the other methods while being less computationally expensive. The nodes estimate the frequency of tones they detect and store them in a list, from which they find the difference of the sequential values to calculate the bandwidths.

The program simulates the down-converted two-tone pulse in a fully connected network to characterize the system's behavior through simulation. Then, it lets the mixing matrix place a

link connection and proper weight to these values when performing the proper multiplication. As explained in section 2.3.2, equation 2.7, each internode syntonization link has its own independent AWG noise, which leads to slightly different estimates. Extending the model proposed by [39, 47, 75–78], instead of 2.13, the algorithm is extended to equation 2.14. Where $\mathbf{M}$ is an $n \times n$ matrix holding each node's measured bandwidth in their corresponding columns, and $diag(\cdot)$ is the diagonal function, which extracts only the diagonal elements of $(\cdot)$

$$\begin{bmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_N. \end{bmatrix}$$

Because the simulation considers a fully connected network of measurements, $\mathbf{M}$ is a nonzero matrix, and through the mixing matrix synthetically it maintains or drops the communication links. For the simulation, $T(k+1)$ is a one-dimensional array that holds the values of the next iteration's bandwidth, and the process is repeated again until convergence is achieved. Equation 2.14 describes the system-level syntonization. At the node level, this is equivalent to performing a dot product between the corresponding rows of $\mathbf{W}$ and $\mathbf{M}(k)$, and the results become the updated bandwidth for the next iteration.

$$f(k+1) = \mathbf{W} \times f(k) \tag{2.13}$$

$$T(k+1) = diag(\mathbf{W} \times \mathbf{M}(k)) \tag{2.14}$$

The algorithm is executed until a stopping criterion is met which depends on a case by case scenario depending on the system's capabilities.

2.6 Simulation Results

The results of averaging 50 Monte Carlos simulation over 50 pulse duration from $10\mu S$ to $100\mu S$ are shown in Figure 2.12, Figure 2.13, and Figure 2.14 with four, five, and six node systems respectively. These results show that increasing the pulse duration and system connectivity greatly impacts the system's performance. For this simulation, the stopping criterion was decided from

the average consensus algorithm iteration counter with a limit of 100 iterations. Then, the last 10 iteration's residual values were averaged and presented.

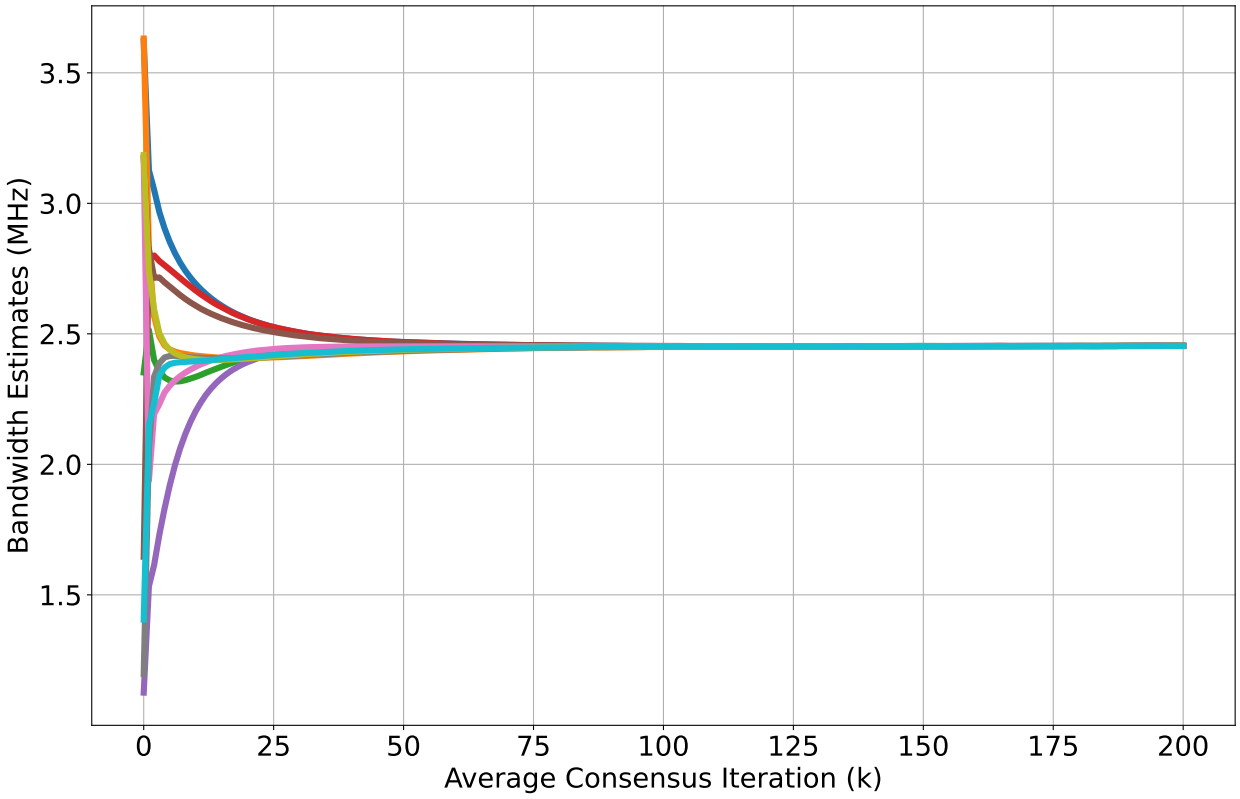

Figure 2.10 Sample average consensus experiment of simulations 100^{-}sec

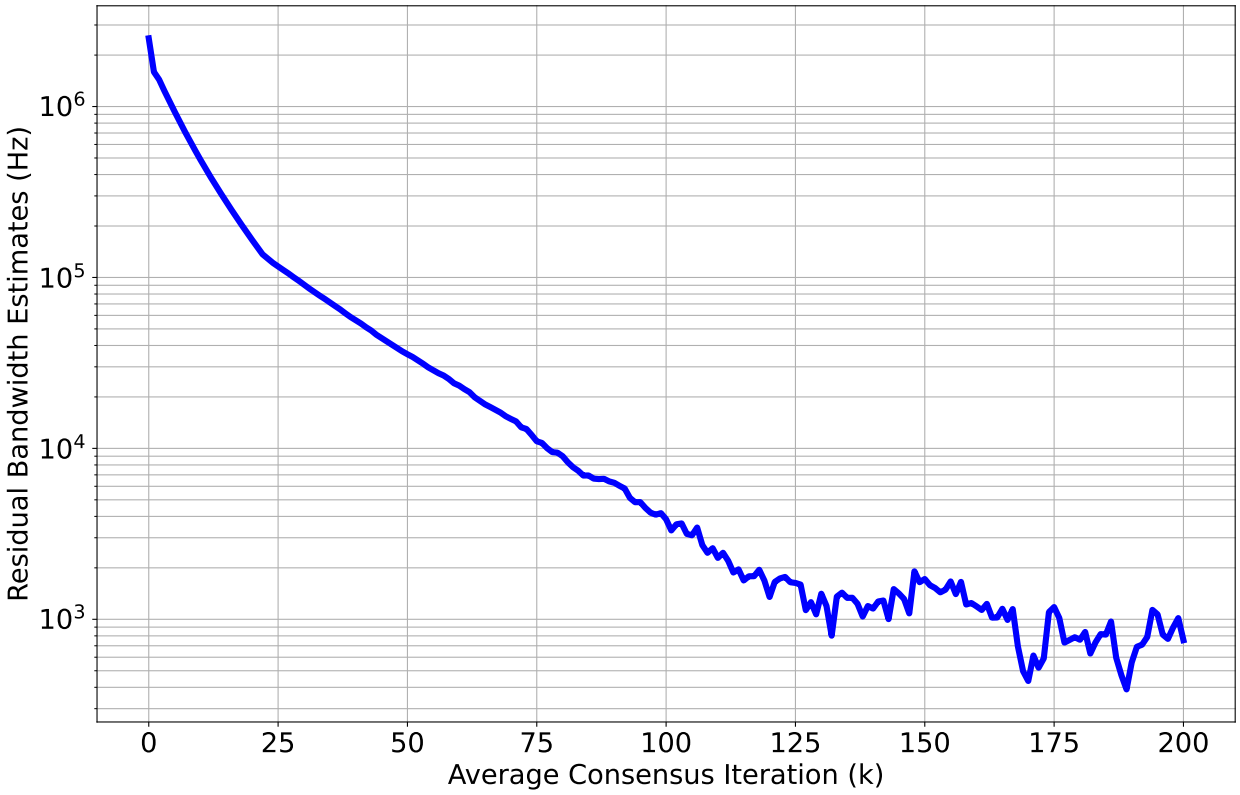

Figure 2.11 Average residual bandwidth at consensus over multiple pulse durations for a four node system with pulse duration of 100^{-}S

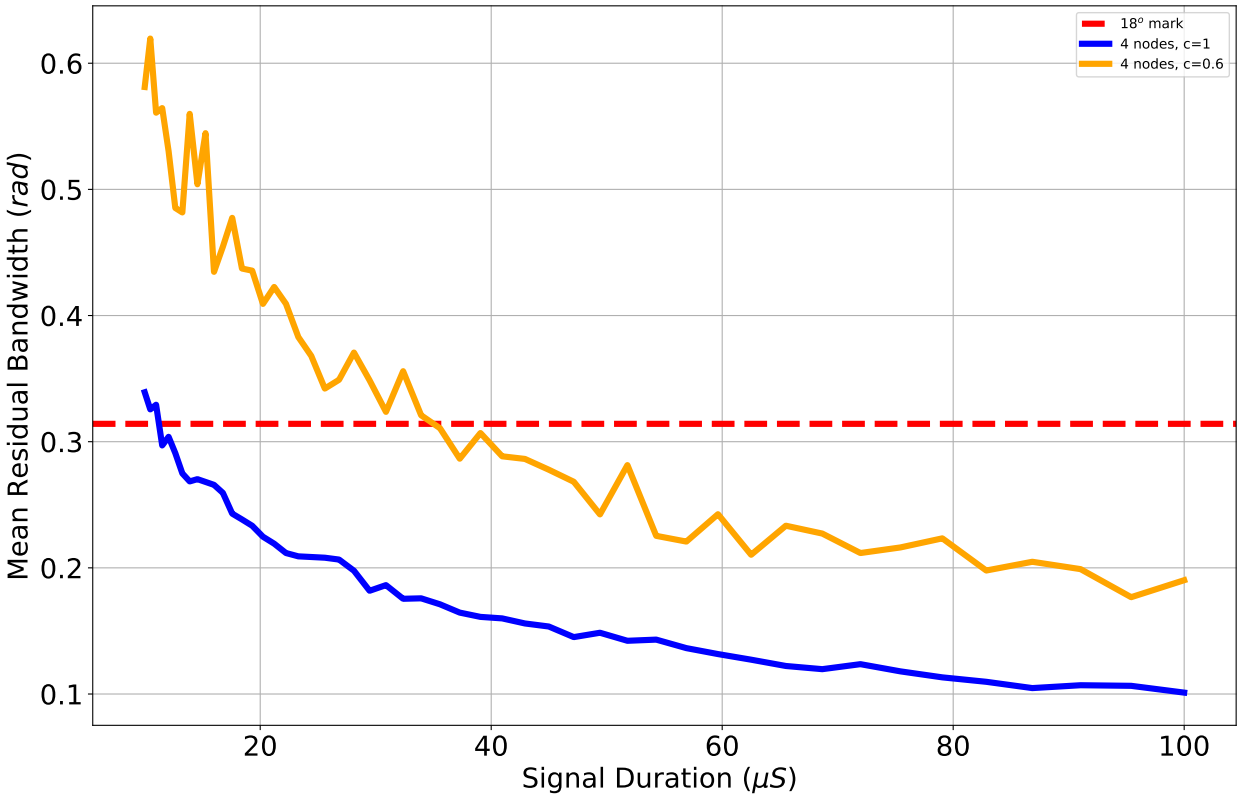

Figure 2.12 Plot of sweeping 50 pulse duration between 10^{-}S to 100^{-}S over 50 Monte Carlos simulation of 4-node system with connectivities of 66% and 100%

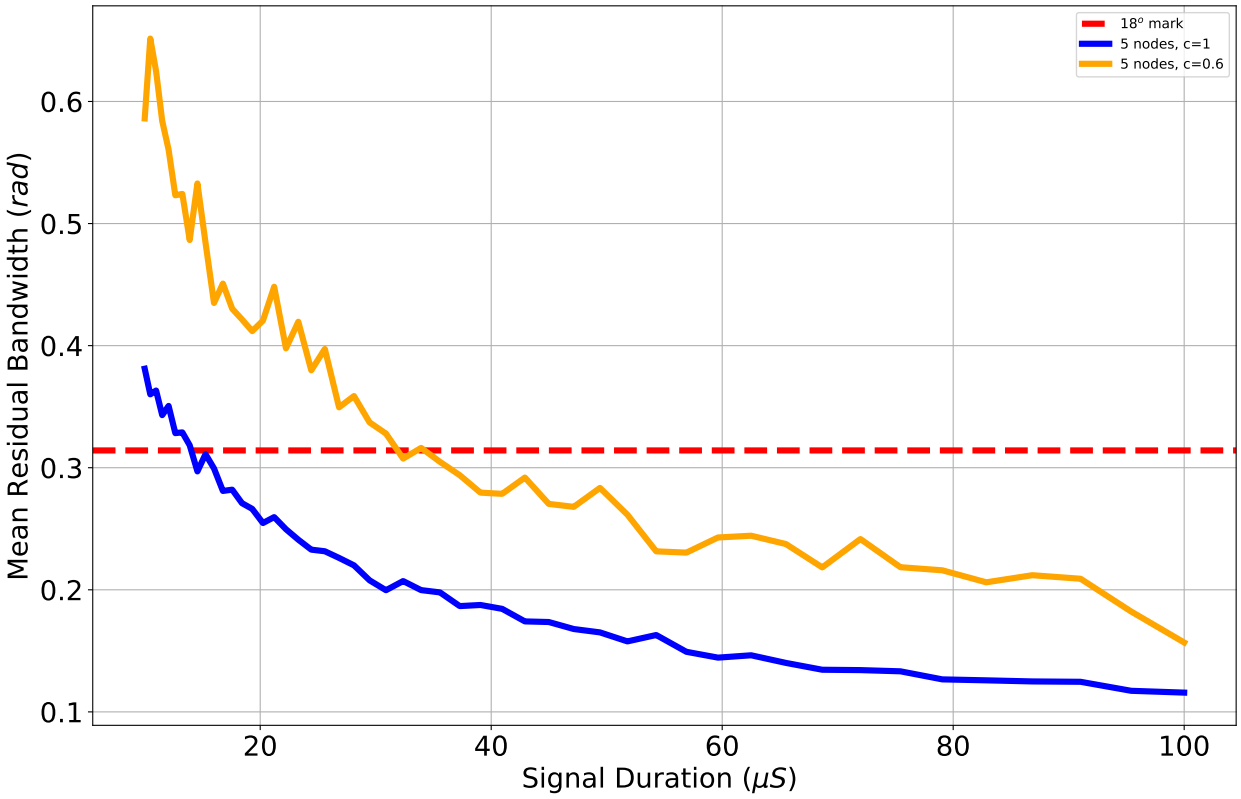

Figure 2.13 Average residual bandwidth at consensus over 50 pulse duration, for two different topologies composed of 5 nodes

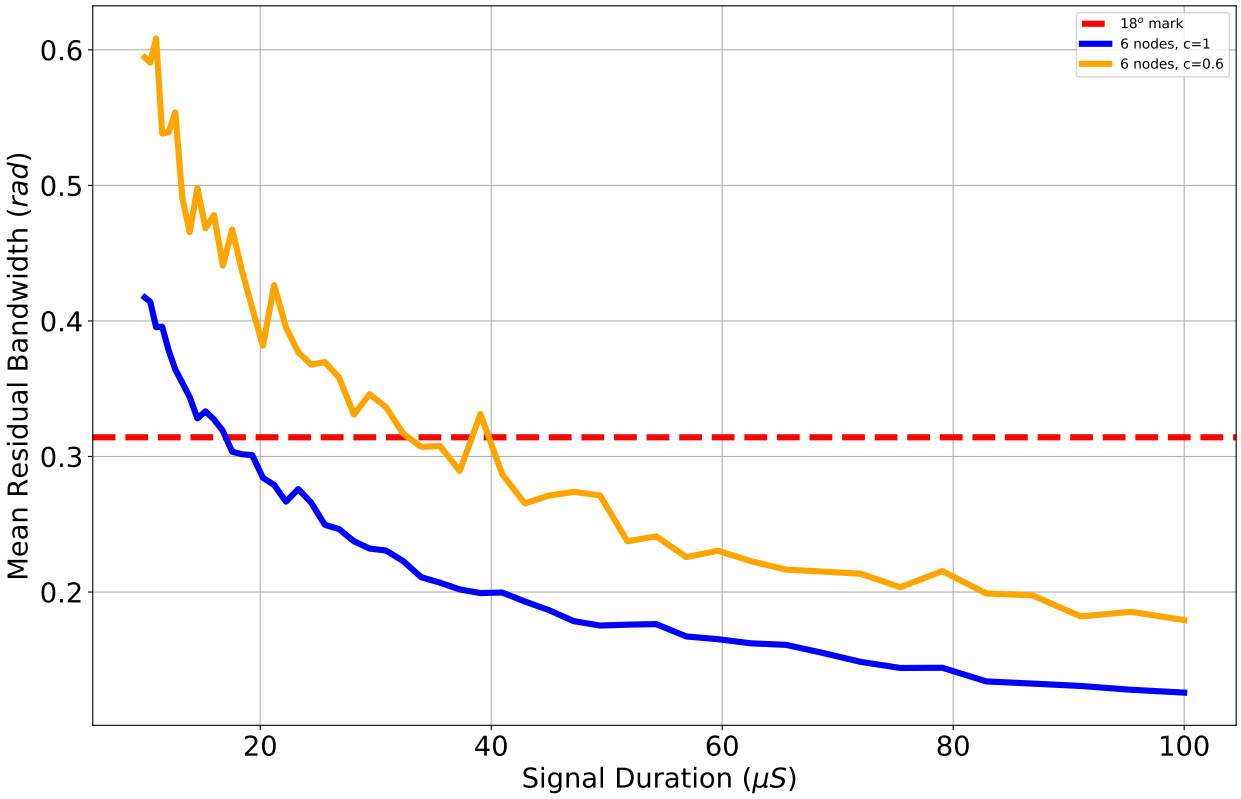

Figure 2.14 Average residual bandwidth at consensus over 50 pulse duration, for two different topologies composed of 6 nodes

2.7 Conclusion

From the Monte Carlos simulations, it can be concluded that for a small network, no larger than six nodes, the minimum pulse duration needed to ensure stability less than the $18°$ needs to be longer than $40\,\mu s$, with a capture window of $120\,\mu s$. It is important to reiterate that these simulations are valid for small-size networks, each signal would have $30\,dB$ SNR, but the SNR as shown in Figure 2.10, and Figure 2.11, where the network is composed of over 10 nodes and signal duration of $100\,\mu s$, the residual error yields a phase offset of approximately $36°$.

CHAPTER 3

SYSTEM DESIGN

3.1 Introduction

In this chapter, the SDRs experimental implementation to perform a wavelength level of stability is discussed. The frequency stabilization can be divided into two significant components, the system as a whole and the individual nodes. The node level is divided into the hardware's functionality, and its corresponding software. It is important to understand the building block of the system, how they perform individually, collectively and as a whole unit. These are the pillars of the innovation provided by this project.

3.2 System Architecture

No other research group or industry has fully explored the unique characteristic of distributed phased arrays: the open-loop and decentralized architecture concept. This innovation will permit the technology to be mounted on spatially independent moving vehicles that can be sent to remote areas where both sensing and communication applications are crucial for performing tasks, but control of the units is limited.

As explained before, the system can be described as a graph with vertices corresponding to the graph's nodes and links connecting them. To develop distributed and open-loop capabilities, the graph must be strongly connected; in other words, a path of links and nodes exists that allows information to propagate to all the nodes, flooding the network. There are two ways to achieve this, and the most convenient way is to assume the links are undirected, which means the nodes have bidirectional communication. To achieve bidirectional communication, it is necessary to share the resources available, and the use of OFDM significantly facilitates this. Time-division multiplexing is another option, but this project intends to demonstrate full timing independence while performing signal syntonization. Another approach is to use a different gossip-based network flooding approach [34], such as push-sum protocol [47], which allows the network to have directed communication links. Figure 2.1 is an example of a decentralized and strongly connected

network. The advantage of this kind of network is that information sharing occurs at the internode level without external feedback from third parties. Implicit information sharing is also enabled, enhancing the security and resilience of the system, which aligns with the project's goal.

3.3 System Components

The system is composed of multiple nodes with simultaneous transmission and reception capability. For the physical system SDRs take the place of the nodes and the link that allows for inter-node communication are the two-tone signals transmitted wirelessly that implicitly convey the bandwidth information. As an initial step, to demonstrate the viability of the approach taken to stabilize the signal the SDRs are all connected through $10\,\mathrm{Gbit/s}$ Ethernet cables to a single host computer unit running GNURadio Companion in Linux Ubuntu 20.04 operating system. The reason for connecting the SDRs to a single computer is to partially address the timing mismatch of the SDRs.This way they all activate, transmit, and receive relatively almost at the same time.

3.4 Data Flow and Storage

The system is initialized with each node agreeing upon the portion of the spectrum that they will utilize to transmit their signals and identify their neighbors. The SDRs can be close to each other and at line of sight. When necessary, they can be synthetically dropped even when maintaining the spatial proximity for extreme cases where the estimates are unreliable. As a rule of thumb, the nodes should transmit within the same instantaneous bandwidth that the nodes can receive to avoid random oscillator drifts when updating the local oscillators to search for signals through the spectrum. Also, the signals must be independent of each other. Hence, a wide enough separation, band guard, exists between signals and coupled tones. Assuming the system has obtained a coarse time alignment that allows each node to transmit relatively simultaneously, the time delays related to the positioning of the nodes can be neglected. In the cases where the signals that are being transmitted are pulsed, then a tighter condition for time alignment is necessary, but if all the nodes are transmitting continues waves (CW) then time alignment is only required for updating the tone's frequencies in the transmitter end, but not as crucial in the receivers side of the node.

The receiving nodes do not explicitly receive the tone values or the bandwidth at which

the rest of the network is operating because of the difference between the nominal frequency value inputted by the software and the actual transmission [30], hence the receiving nodes implicitly receive the transmitting frequencies and are responsible of down-converting, detecting, identifying the transmitter, estimating the received frequencies, calculate the bandwidth, compute their local average and update for the next cycle of transmission.

This cycle of signal transmission, reception, estimation, average calculation, and transmit updating continues indefinitely, syntonizing the local node transmission with the other nodes in the system. Figure 3.1 summarizes the flow of the data and decision making by each node that cumulatively achieves consensus at the node level and convergence at the system level.

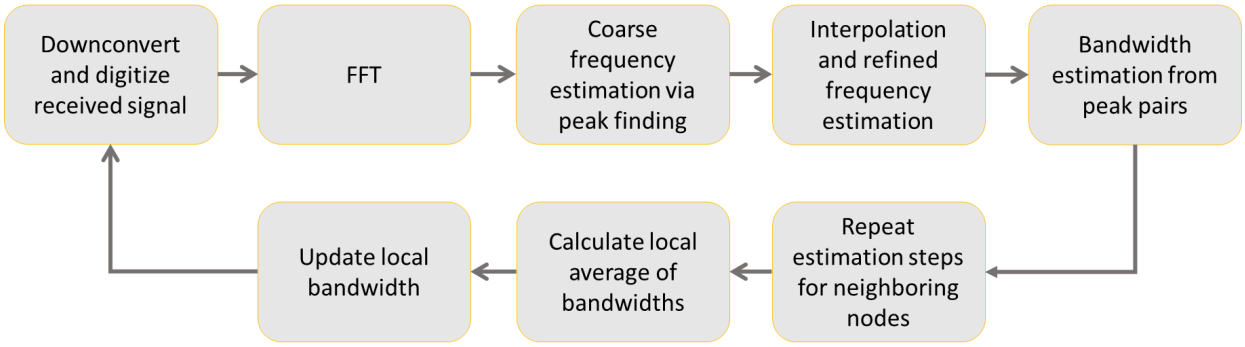

Figure 3.1 Block diagram of information flow at a local node level.

Given the SDRs' memory capabilities, there are two options for deciding the estimated value used for calculating the local average: either the input value stored in memory or its own self-estimated bandwidth value. For the sake of spectral awareness, it was decided to use the current transmission estimation value but store the previous transmission estimation locally for self-reporting and offline analysis.

3.5 Technologies and Tools

As stated multiple times previously, this project aims to demonstrate the capabilities and limitations of designing a decentralized distributed phased array using software-defined radios that can perform syntonization purely through software using signal processing approaches. This is why the technology used is SDRs that run solely on software and omnidirectional antennas to

communicate locally among the SDRs.

3.5.1 Node Design

In the effort to demonstrate the capabilities of decentralized open loop network antenna array stabilization, the average consensus algorithm was validated using National Instrument (NI) Ettus Research USRP X310 software defined radios [79], with NI UBX160 daughter-boards [80]. The reason for the use of these software-defined radios is the multiple methods of programming and interfacing to software the NI technology allows, ranging from high-level programming with LabView, MATLAB Simulink, and GNU Radio Companion with Python to low-level programming with Verilog.

Figure 3.2 National Instrument's picture of USRP X310 Software Defined Radio.

We have decided to use the GNU Radio block diagram development environment, which allows high-level programming languages such as C++ and Python to create the flow diagram that manipulates data for processing and takes advantage of the plethora of signal processing libraries these languages possess.

3.5.2 Antenna Selection

The Pulse Electronics Multi Band Swivel Mount Dipole SPDA24700/2700 antennas were used to verify the validity of the proposed approach. Each SDR has two antennae connected to their corresponding daughter board, and each daughter board is assigned the task of either trans-

Figure 3.3 National Instrument's picture of UBX160 daughterboard for X310 SDR.

mitting or receiving. These are omnidirectional antennae; therefore, if the SDRs are placed close to each other but far away from the near field of the antenna and the transmit frequencies, then all the SDRs can communicate with each other, enabling a completely connected network to start with, which estimated values can be zeroed out to generate smaller networks and different network architectures synthetically. Another reason, besides being omnidirectional, to choose these antennas for the setup is the bandwidth of operation of the antennas, which allowed for proper operation with the X310 SDRs in a lab environment within Michigan State University.

3.6 Algorithms and Methods

The average consensus method of frequency stabilization described in section 2.2 is utilized, and the different bandwidths calculated previously are averaged over the SDRs' local mean. This means that the values of each bandwidth are properly mapped to their corresponding

transmitting SDR and assigned the necessary weight, which is multiplied and consequently added to compute the node's local average. This operation is performed locally at the node level, and as a composite, the system approaches a single bandwidth value.

3.7 Scalability and Performance

The reason for taking the approach of orthogonal frequency division multiplexing is to be able to take advantage of spectral orthogonality to identify the signals and their corresponding transmitting nodes, instead of time orthogonality, which would require a predetermined schedule and protocol for new nodes to enter the network, as well as drop from the system. As far as the author of this thesis is aware, techniques such as those demonstrated in [81] require a syntonized system, which has only been achieved by connecting the SDRs through cables or using self-looped external hardware.

As a proof of concept, we have experimented with a relatively small-scale network of nodes. To sustain the assumption of coarse time alignment, we have utilized a centralized computation center as depicted in section 3.2. The connection of four nodes to the same computation center leaves the distribution of computation resources to the computer architecture, limiting the system's scalability. To allow for full scalability of the network and system, it would be necessary to provide each node with its computational unit or directly program the field-programmable gate array (FPGA) to perform frequency estimation, average calculation, and transmit frequency update. This means that the node's clock needs to be synchronized adequately to achieve complete independence.

3.8 Testing and Validation

To define convergence, interest in the system-level performance is important. Taking as reference [39, 40, 47, 52] the desired accuracy of the should be less than 18°. The conversion of the tolerable error for $90\,\%$ relative gain, of 18° is elaborated more in [7, 40]. That is why although the goal of achieving a residual error of less than $10\,\mathrm{Hz}$ is a significant milestone that can mimic monolithic arrays, it is possible with only under pulse duration that is longer than $150\,\mu\mathrm{s}$. To validate the system's performance, it was decided to use the self-reported values and estimate

Figure 3.4 Experimental setup composed of four SDRs, with omnidirectional antennas, connected to a single computing unit. The system was designed to wirelessly syntonize two-tone signals without node synchronization. A coarse synchronization is achieved by connecting the SDRs to the same computer and daisy-chaining a reference clock.

the residual error against the mean bandwidth value. Figure 3.4 is a picture of the setup of the four SDRs with their corresponding antennas arranged in a square for line-of-sight communication among themselves. The results of the different network architectures are presented in the following chapter, Chapter 4, with a comparison against those simulated in the previous chapter, Chapter 2 .

3.9 Conclusion

The proposed system design for syntonizing a decentralized, distributed phased array composed of SDRs is unique in its class because it does not require any external hardware connected to the SDRs, and it aims to achieve syntonization solely through signal processing algorithms. The integration of all system components enables the validation of the proposed methodologies. Under the constraints presented and addressed in this section, the system may be operable and demonstrate promising results that need further validation.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Introduction

The intent of Chapter 4 is to present the experimental results of the experiments performed using the hardware setup and design presented in Chapter 3 in a controlled lab environment as a proof of concept and compare these results with the simulations performed in Chapter 2. Under these conditions, the experiment validates the minimum signal duration needed and is viable to achieve the 18° residual phase error. Also, this chapter includes a comparison of the state-of-the-art technology and the proposed approach.

4.2 Results

4.2.1 Fully Connected 4-Node Network With No Frequency Hopping

The first network experimented on was a fully connected network of four SDRs. This network has a connectivity of $100\,\%$; in essence, every node communicates directly with the other nodes and itself. Theoretically, the system syntonizes perfectly to a single value if the nodes have no local estimation error. In reality, each node is independently estimating the two-tone signal bandwidths of the other nodes, and inherently, these estimations have errors associated with each tone estimation that add up to the bandwidth estimation and impact the calculation of the local average value of the node, preventing the system from agreeing upon to a single value. This is demonstrated in the inlet of Fig.4.1, where the estimated transmission bandwidths are all oscillating around the mean value of the initial bandwidths after $k = 1$. For this experiment, the measurement matrix, $\mathbf{M}$, was not modified to follow any hopping scheme. The average of 25 experiments with four different signal durations is presented in table 4.1, summarizing the average central frequency offset (CFO), accuracy with respect to the carrier frequency, and phase error.

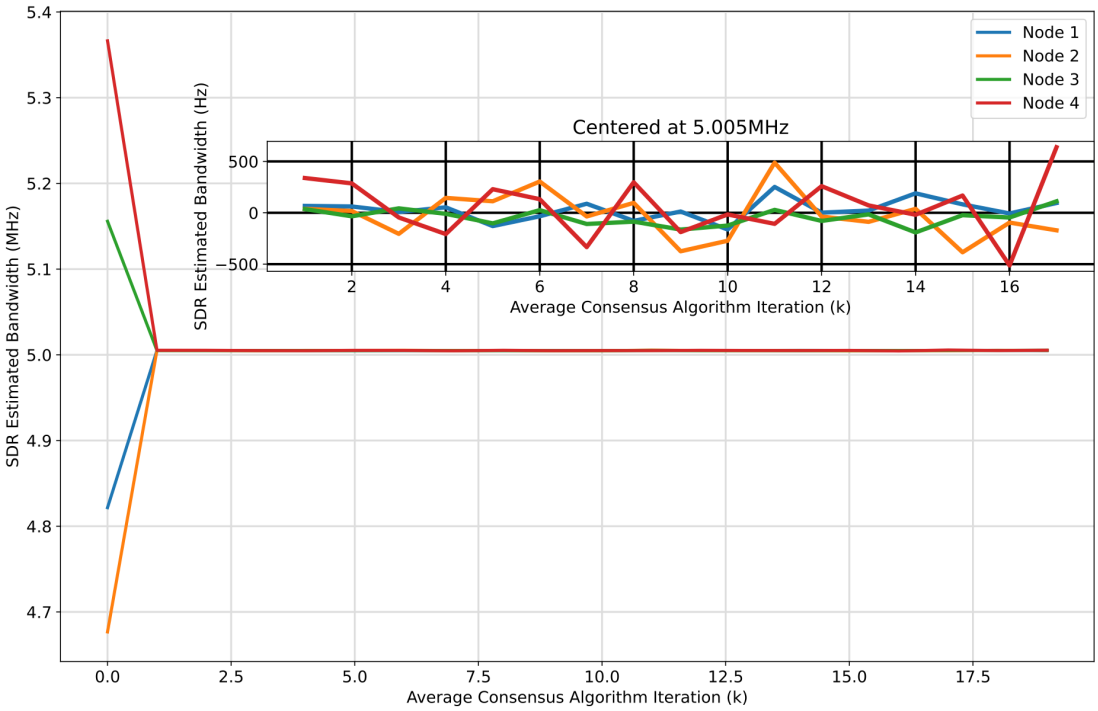

Figure 4.1 Sample real-time experiments with no frequency hopping, fully connected network with signals' duration of $50\,\mu s$

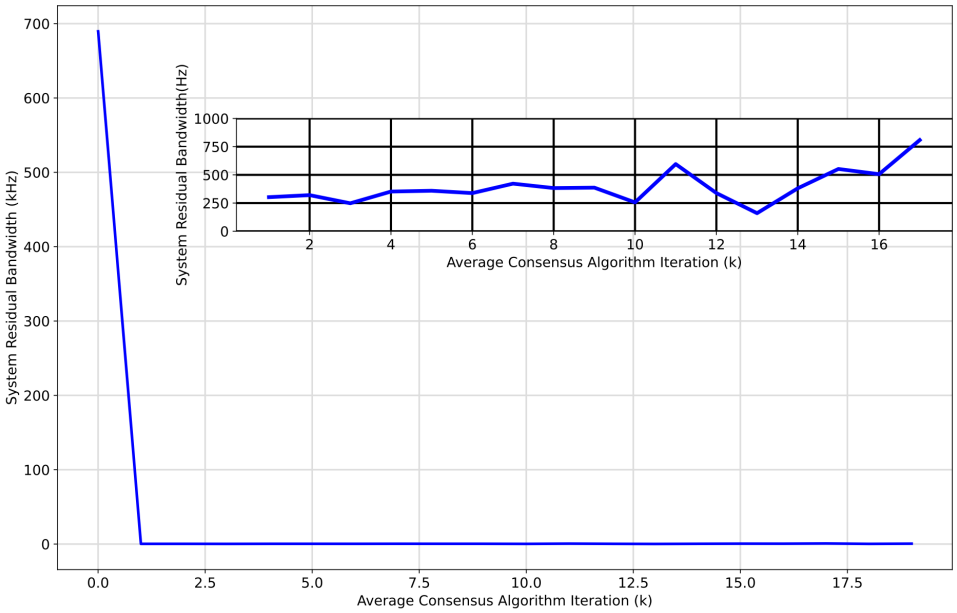

Figure 4.2 Sample real-time experiments residual error with no frequency hopping, fully connected network with signals' duration of $50\,\mu s$

Table 4.1 Average CFO: 25 real-time experiments with no frequency hopping, fully connected network

Pulse duration (μs)	CFO(Hz)	Accuracy (ppb)	Phase Error (rad)
$\tau = 20$	855.58	407	0.107515
$\tau = 30$	495.68	236	0.093443
$\tau = 40$	354.60	169	0.089120
$\tau = 50$	245.68	117	0.077183

4.2.2 Loop Connection 4-Node Network With No Frequency Hopping

The second network experimented on was a looped network of four SDRs. This network has a connectivity of $66.66\,\%$; the nodes communicate with their nearest neighbor and themselves. For this scenario, each node independently estimates the two-tone signal bandwidths, and inherently, there is an estimation error associated with each tone estimation that adds up to the bandwidth estimation and impacts the local arithmetic mean calculation. The simultaneous presence of multiple two-tone signals in the same part of the spectrum induces an estimation error larger than the CRLB, as explained in 2.4.3. Even though the error is present in each bandwidth estimation that affects their local average, the bandwidth values remain close to each other as shown in Fig.4.3 and in more detail in the sample plot of the residual values, Fig.4.4. As explained in sec.4.2.1, because no frequency hopping scheme has been implemented yet hence, the measurement matrix, **M**, is not changed from one iteration to another.

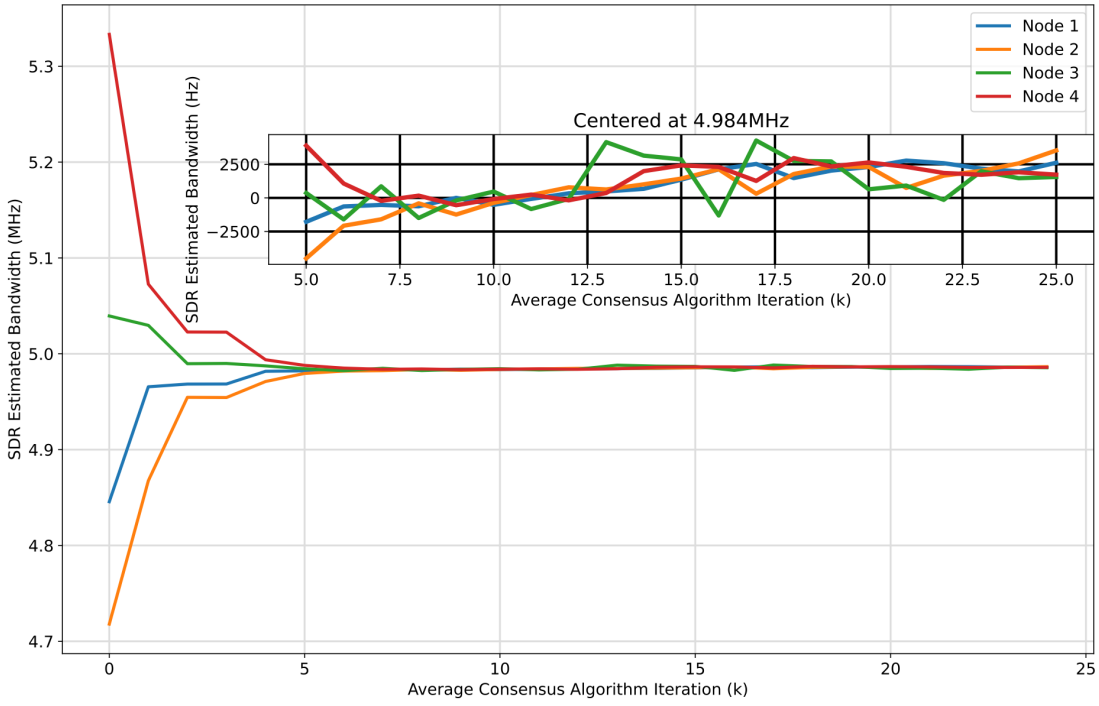

Figure 4.3 Sample real-time experiments with no frequency hopping, looped network with signals' duration of $50\,\mu s$

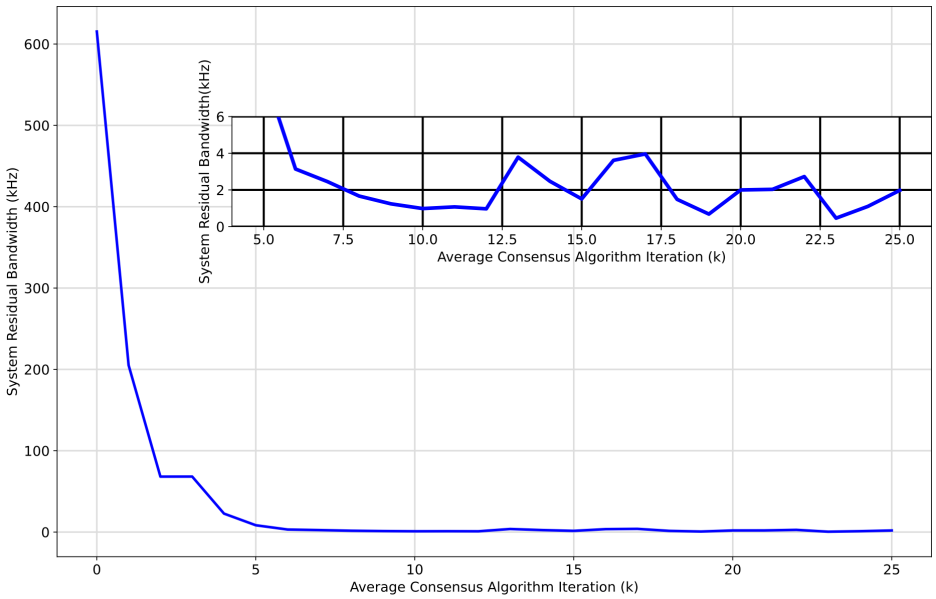

Figure 4.4 Sample real-time experiments residual error with no frequency hopping, looped network with signals' duration of $50\,\mu$s

Table 4.2 Average CFO: 25 Real-time experiments with no frequency hopping of a looped network

Pulse duration (μs)	CFO(Hz)	Accuracy (ppb)	Phase Error (rad)
$\tau = 20$	3460.50	1648	0.447425
$\tau = 30$	1541.40	734	0.290547
$\tau = 40$	935.03	445	0.234998
$\tau = 50$	877.75	418	0.275753

4.2.3 Fully Connected 4-Node Network While Frequency Hopping

The third network experimented on was a fully connected network of four SDRs. This network has a connectivity of $100\,\%$; in essence, every node communicates directly with the other nodes and itself. The mixing matrix, **W**, has all the elements with a weight of 0.25; the measurement matrix, **M**, has all the entries with their corresponding measurement. For this set of experiments, the measurement matrix changed according to the schedule described in fig.2.3. Ideally, mapping the estimated bandwidth to the schedule, the system syntonizes perfectly to a single value if the nodes have no local estimation error. The impact of tones and bandwidth estimation is demonstrated in the inlet of Fig.4.5, where the estimated transmission bandwidths are all oscillating around the mean value of the initial bandwidths after $k = 1$. The average of 25 experiments with four different signal durations is presented in table 4.1, summarizing the average central frequency offset (CFO), accuracy concerning the carrier frequency, and phase error.

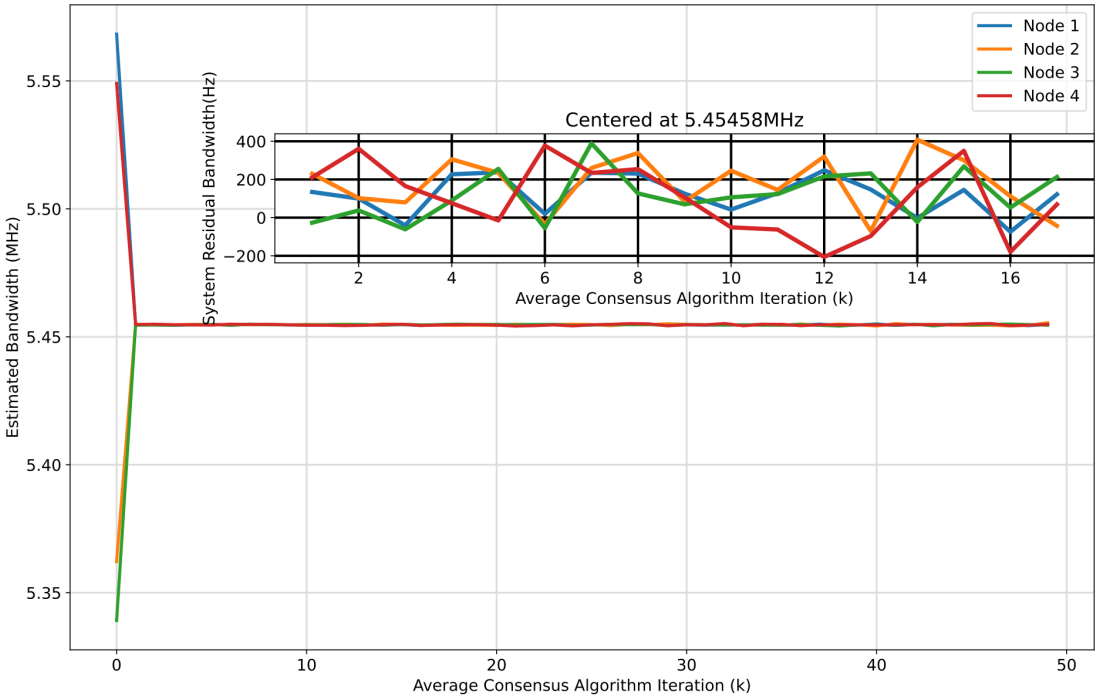

Figure 4.5 Sample real-time experiments while frequency hopping, fully connected network with signals' duration of $50\,\mu s$

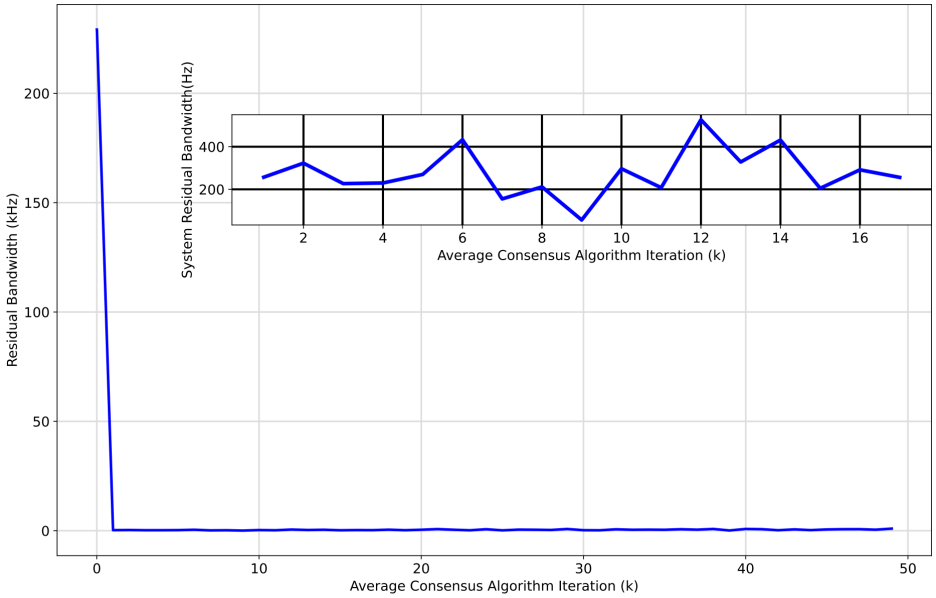

Figure 4.6 Sample real-time experiments residual error while frequency hopping, fully connected network with signals' duration of $50\,\mu s$

Table 4.3 Average CFO: 25 real-time experiments while frequency hopping, fully connected network

Pulse duration (μs)	Connected CFO(Hz)	Accuracy (ppb)	Phase Error (rad)
$\tau = 20$	1458.76	695	1.816941
$\tau = 30$	520.00	248	0.098017
$\tau = 40$	424.36	202	0.106653
$\tau = 50$	247.99	118	0.077908

4.2.4 Loop Connection 4-Node Network While Frequency Hopping

The fourth and final network experimented with was a looped, connected network composed of four SDRs, operating in frequency hopping mode. This network has a $66.66\,\%$ connectivity. After looking into the data, it was brought to the author's attention that the behavior of the experiment's behavior is not the one expected because the mapping of the measurement matrix, **M**, and connectivity matrix, **W**, did not happen simultaneously according to the schedule depicted in fig2.3. The connectivity matrix trailed the measured matrix by one iteration update. Hence, the behavior described in fig4.7 is different than the behavior in fig4.1, and consequently, the behavior in fig4.8 is different from the behavior in fig4.4. Even though this error persists in every iteration of this experiment, the network happens to approach the global mean value at the beginning, $k = 0$. Still, contrary to the non-hopping experiment, the bandwidths do not follow each other but remain close among themselves.

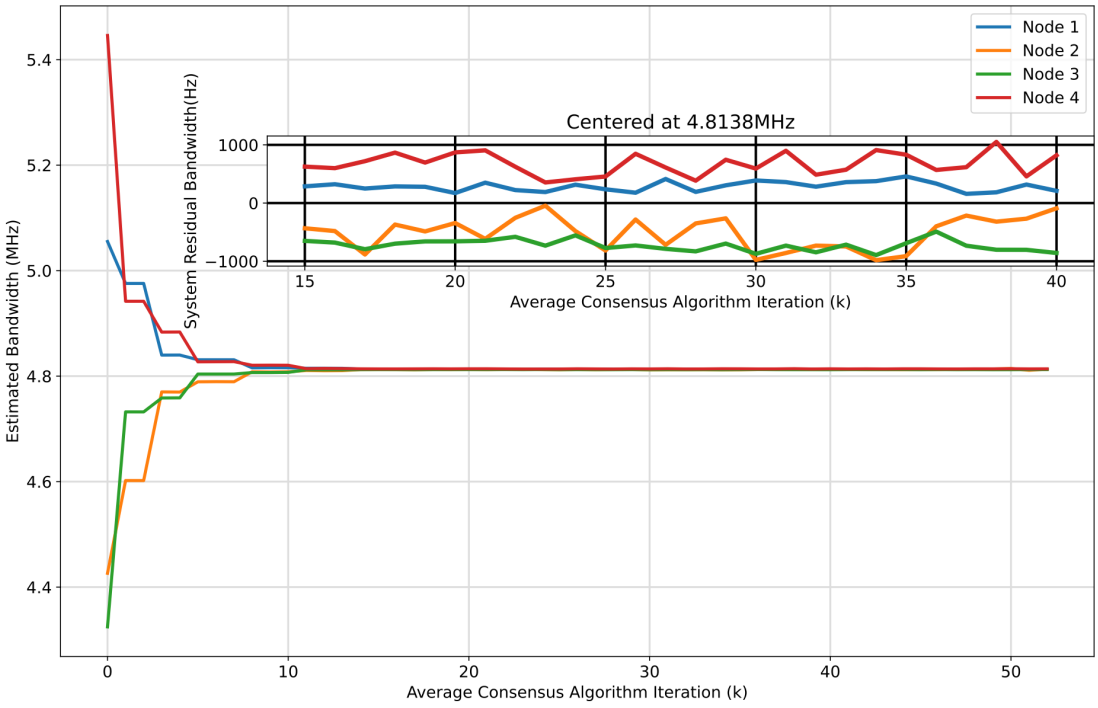

Figure 4.7 Sample real-time experiments while frequency hopping, looped network with signals' duration of $50\,\mu s$

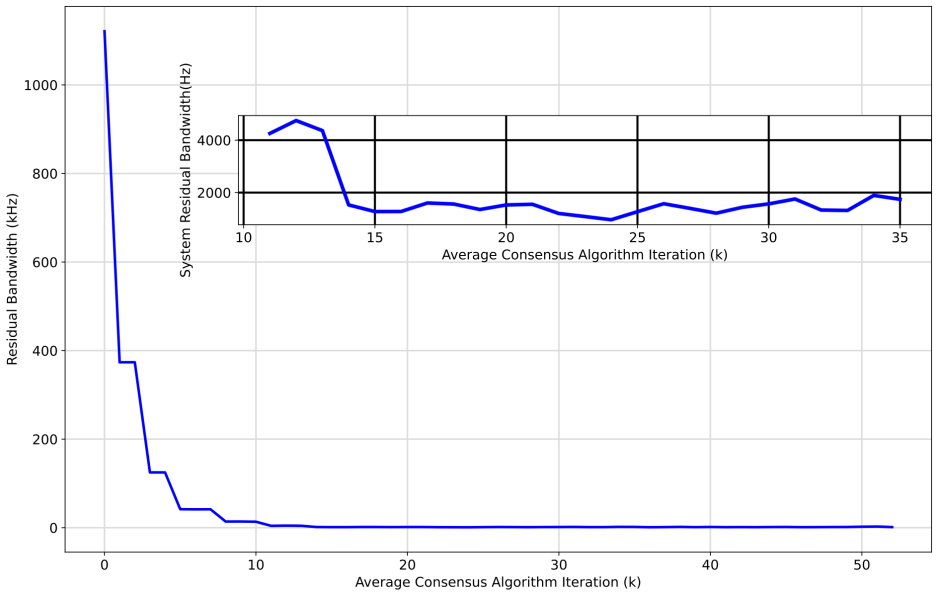

Figure 4.8 Sample real-time experiments residual error while frequency hopping, looped network with signals' duration of $50\,\mu$s

Table 4.4 Average CFO: 25 real-time experiments while frequency hopping, looped network

Pulse duration (μs)	Connected CFO(Hz)	Accuracy (ppb)	Phase Error (rad)
$\tau = 20$	1874.58	893	0.235566
$\tau = 30$	1606.40	765	0.302799
$\tau = 40$	1290.00	614	0.324212
$\tau = 50$	853.08	406	0.268003

4.3 Discussion

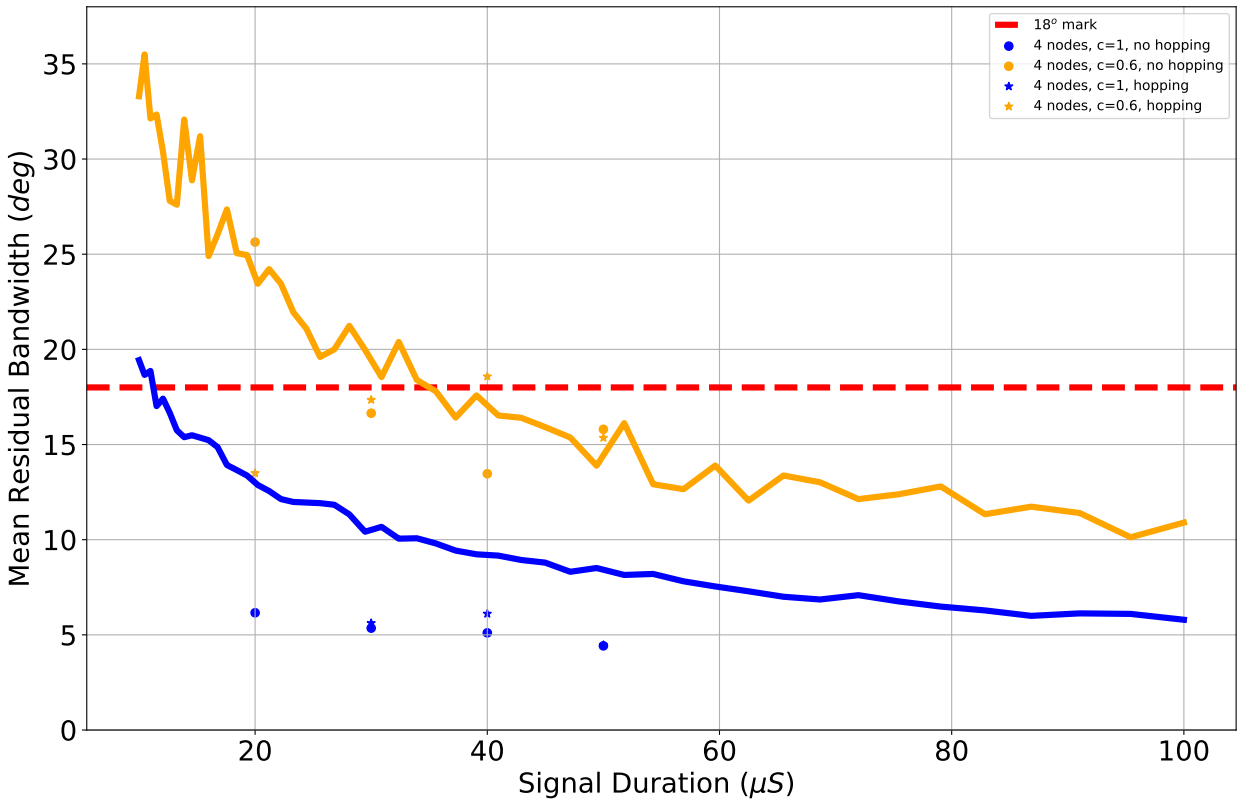

Figure 4.9 Experimental and simulation results.

Figure 4.9 summarizes the results obtained during the experiment and are overimposed with the simulated results, allowing for easy comparison between the theoretical and experimental values. As expected from the simulation, longer signals can achieve better estimation, but longer signals may result in higher residual phase error, given the instability of the oscillators. From simulations, it is clear that fully connected networks can go past the 18° set threshold with signals with durations greater than $10\,\mu s$ while looped networks need to last more than $30\,\mu s$. Experimental values were able to surpass the 18° marker from $20\,\mu s$ disregarding the type of network. This could have been because the initial errors introduced during the experiment differed from those introduced for the simulations, given that the simulations' initial errors were introduced from a more significant standard deviation from a pseudo-random number generator. Although the experimental results are better than the simulated values, this can be attributed to the randomness of the

two-tone signal's bandwidth for each scenario. This proves that the estimation accuracy from the beginning of the average consensus algorithm is crucial for the result of the consensus problem.

The results obtained from the experiments open the door for new research, such as incorporating simultaneous signals syntonization with synchronization, enabling other waveforms' syntonization, and comparing them to figure out which one allows for better parameter estimation and, consequently, more stable syntonization. Also, it would be interesting to experiment and understand dynamic networks that incorporate or disable links depending on the quality of the communication links among nodes.

4.4 Conclusion

In summary, the simulation and experimental results prove that to syntonize the two-tone signals' bandwidth, it is essential to have at least signal durations greater than $40\,\mu\text{s}$. These results provide the necessary evidence to support this thesis while bringing new considerations regarding the scalability of this approach as the electromagnetic spectrum becomes more crowded and the signals interfere more and more with each other. The implications of these findings and their alignment with existing literature should be explored in greater depth in the future.

CHAPTER 5

CONCLUSION

An attempt at two-tone bandwidth syntonization was presented to align the signals of a distributed phased array. A small, four-SDR experimental system is presented, and the self-evaluated frequency deviations are reported. This is the first attempt to achieve two-tone bandwidth syntonization of pulsed signals solely through signal processing on off-the-shelf SDRs for a fully distributed system without compensating for time misalignment. In the system, where all SDRs are connected to the same computer and detect signals from all SDRs, experiments are conducted with both a fully connected network and a looped network, presenting multiple pulse durations. Upon receiving the superposition of its neighboring node signal, down-converting and digitizing the analog signals, the SDRs would estimate each tone frequency and bandwidth. Once the SDRs had a list of detected bandwidths, they would calculate the weighted average of the bandwidths, which would become their next iteration's transmission bandwidth, until they asymptotically achieved a global average. This experiment aimed to achieve an integral phase offset of less than 18° solely due to the frequency mismatch. Both in simulation and experimentally, this was achieved.

Although the experimental measurements presented in this thesis were performed at a carrier frequency of $2.1\,\mathrm{GHz}$, it can be extended to higher carrier frequency signals and longer pulse durations given that the SDRs can receive longer signal duration if using an input data buffer, contrary to the work presented here. If the experiment is conducted at higher frequencies and directly at RF, then the amount of unnecessary data could exacerbate the problem of memory and computing power that limits the system's performance and improvement. This project demonstrates that, in practice, syntonization remains independent even when there is no time alignment in the system. Orthogonal signals can be used to synchronize the bandwidths of two-tone signals for distributed phased array alignment. Further investigation is essential for developing an independent computation and communication schedule for the nodes comprising the distributed phased array to fully realize the potential of distributed phased array systems.

BIBLIOGRAPHY

[1] Ying Tan and Zhong yang Zheng. Research advance in swarm robotics. *Defence Technology*, 9(1):18–39, 2013.

[2] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial isystems*. Oxford University Press, New York, 1999.

[3] Suhanya Jayaprakasam, Sharul Kamal Abdul Rahim, and Chee Yen Leow. Distributed and Collaborative Beamforming in Wireless Sensor Networks: Classifications, Trends, and Research Directions. *IEEE Communications Surveys & Tutorials*, 19(4):2092–2116, 2017. Conference Name: IEEE Communications Surveys & Tutorials.

[4] J. C. Merlano-Duncan, J. Querol, A. Camps, S. Chatzinotas, and B. Ottersten. Architectures and synchronization techniques for coherent distributed remote sensing systems. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 8875–8878, 2019.

[5] Samuel Prager, Mark S. Haynes, and Mahta Moghaddam. Wireless Subnanosecond RF Synchronization for Distributed Ultrawideband Software-Defined Radar Networks. *IEEE Transactions on Microwave Theory and Techniques*, 68(11):4787–4804, November 2020. Conference Name: IEEE Transactions on Microwave Theory and Techniques.

[6] Divya Pandey and Vandana Kushwaha. An exploratory study of congestion control techniques in wireless sensor networks. *Computer Communications*, 157:257–283, 2020.

[7] Jeffrey A. Nanzer, Serge R. Mghabghab, Sean M. Ellison, and Anton Schlegel. Distributed Phased Arrays: Challenges and Recent Advances. *IEEE Transactions on Microwave Theory and Techniques*, 69(11):4893–4907, November 2021. Conference Name: IEEE Transactions on Microwave Theory and Techniques.

[8] Raghuraman Mudumbai, D. Richard Brown Iii, Upamanyu Madhow, and H. Vincent Poor. Distributed transmit beamforming: challenges and recent progress. *IEEE Communications Magazine*, 47(2):102–110, February 2009. Conference Name: IEEE Communications Magazine.

[9] Harry L Van Trees. *Detection, estimation, and modulation theory, part IV: detection, estimation, and linear modulation theory*. John Wiley & Sons, 2002.

[10] Wen-Qin Wang. Carrier Frequency Synchronization in Distributed Wireless Sensor Networks. *IEEE Systems Journal*, 9(3):703–713, September 2015. Conference Name: IEEE Systems Journal.

[11] Waseem Khan, Ijaz Mansoor Qureshi, and Sarah Saeed. Frequency Diverse Array Radar With Logarithmically Increasing Frequency Offset. *IEEE Antennas and Wireless Propagation Letters*, 14:499–502, 2015. Conference Name: IEEE Antennas and Wireless Propagation Letters.

[12] Rayan Merched El Masri, Stephan Sigg, and Michael Beigl. An asymptotically optimal approach to the distributed adaptive transmit beamforming in wireless sensor networks. In *2010 European Wireless Conference (EW)*, pages 511–518, April 2010.

[13] Yuan Ding, Vincent Fusco, and Junqing Zhang. Phase error effects on distributed transmit beamforming for wireless communications. In *2017 11th European Conference on Antennas and Propagation (EUCAP)*, pages 3100–3103, 2017.

[14] Sami M. Lasassmeh and James M. Conrad. Time synchronization in wireless sensor networks: A survey. In *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, pages 242–245, March 2010. ISSN: 1558-058X.

[15] Jiaxin Lu, Feifeng Liu, Jingyi Sun, Yingjie Miao, and Quanhua Liu. Distributed Radar Robust Location Error Calibration Based on Interplatform Ranging Information. In *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*, pages 1–5, December 2019.

[16] H. Ochiai, P. Mitran, H.V. Poor, and V. Tarokh. Collaborative beamforming for distributed wireless ad hoc sensor networks. *IEEE Transactions on Signal Processing*, 53(11):4110–4124, November 2005. Conference Name: IEEE Transactions on Signal Processing.

[17] R. Mudumbai, G. Barriac, and U. Madhow. On the Feasibility of Distributed Beamforming in Wireless Networks. *IEEE Transactions on Wireless Communications*, 6(5):1754–1763, May 2007. Conference Name: IEEE Transactions on Wireless Communications.

[18] U. Madhow, D. R. Brown, S. Dasgupta, and R. Mudumbai. Distributed massive mimo: Algorithms, architectures and concept systems. In *2014 Information Theory and Applications Workshop (ITA)*, pages 1–7, 2014.

[19] Chenyu Zuo, Haoge Deng, Jiyan Zhang, and Yuan Qi. Distributed channel estimation algorithm for mmwave massive mimo communication systems. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pages 1–6, 2021.

[20] Wei Yi, Ye Yuan, Reza Hoseinnezhad, and Lingjiang Kong. Resource scheduling for distributed multi-target tracking in netted colocated mimo radar systems. *IEEE Transactions on Signal Processing*, 68:1602–1617, 2020.

[21] Chenguang Shi, Yijie Wang, Sana Salous, Jianjiang Zhou, and Junkun Yan. Joint transmit resource management and waveform selection strategy for target tracking in distributed phased array radar network. *IEEE Transactions on Aerospace and Electronic Systems*, 58(4):2762–2778, 2022.

[22] Tuomas Aittomäki and Visa Koivunen. Target detection and positioning in correlated scattering using widely distributed MIMO radar. In *The 7th European Radar Conference*, pages 403–406, September 2010.

[23] Jia Xu, Xi-Zeng Dai, Xiang-Gen Xia, Li-Bao Wang, Ji Yu, and Ying-Ning Peng. Optimal transmitting diversity degree-of-freedom for statistical MIMO radar. In *2010 IEEE Radar Conference*, pages 437–440, May 2010. ISSN: 2375-5318.

[24] Max Scharrenbroich and Michael Zatman. Statistical MIMO radar experimental results. In *2014 IEEE Radar Conference*, pages 0617–0622, May 2014. ISSN: 2375-5318.

[25] Alexander Grathwohl, Michael Stelzig, Julian Kanz, Patrick Fenske, Andreas Benedikter, Christina Knill, Ingrid Ullmann, Irena Hajnsek, Alberto Moreira, Gerhard Krieger, Martin Vossiek, and Christian Waldschmidt. Taking a Look Beneath the Surface: Multicopter UAV-Based Ground-Penetrating Imaging Radars. *IEEE Microwave Magazine*, 23(10):32–46, October 2022. Conference Name: IEEE Microwave Magazine.

[26] Kun-Yuan Tu and Chia-Shu Liao. Application of anfis for frequency syntonization using gps carrier-phase measurements. In *2007 IEEE International Frequency Control Symposium Joint with the 21st European Frequency and Time Forum*, pages 933–936, 2007.

[27] Raghuraman Mudumbai, Ben Wild, Upamanyu Madhow, and Kannan Ramchandran. Distributed beamforming using 1 bit feedback: from concept to realization. In *Proceedings of the 44th Allerton conference on communication, control and computation*, volume 8, pages 1020–1027, 2006.

[28] Wayes Tushar and David B Smith. Distributed transmit beamforming based on a 3-bit feedback system. In *2010 IEEE 11th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2010.

[29] Kun-Yuan Tu, Fan-Ren Chang, Chia-Shu Liao, and Li-Sheng Wang. Frequency syntonization using gps carrier phase measurements. *IEEE Transactions on Instrumentation and Measurement*, 50(3):833–838, 2001.

[30] D.W. Allan. Time and Frequency (Time-Domain) Characterization, Estimation, and Prediction of Precision Clocks and Oscillators. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 34(6):647–654, November 1987. Conference Name: IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control.

[31] C. Zucca and P. Tavella. The clock model and its relationship with the Allan and related variances. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 52(2):289–296, February 2005. Conference Name: IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control.

[32] Benjamin Peiffer, Sairam Goguri, Soura Dasgupta, and Raghuraman Mudumbai. An approach to Kalman filtering for oscillator tracking. In *2015 49th Asilomar Conference on Signals, Systems and Computers*, pages 261–265, November 2015. ISSN: 1058-6393.

[33] Serge R. Mghabghab and Jeffrey A. Nanzer. Impact of VCO and PLL Phase Noise on Distributed Beamforming Arrays With Periodic Synchronization. *IEEE Access*, 9:56578–56588, 2021. Conference Name: IEEE Access.

[34] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491, 2003.

[35] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, June 2006. Conference Name: IEEE Transactions on Information Theory.

[36] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, 95(1):215–233, January 2007. Conference Name: Proceedings of the IEEE.

[37] Pratik Chatterjee and Jeffrey A. Nanzer. Effects of time alignment errors in coherent distributed radar. In *2018 IEEE Radar Conference (RadarConf18)*, pages 0727–0731, April 2018. ISSN: 2375-5318.

[38] Pratik Chatterjee, Jeffrey A. Nanzer, and Ming Yan. Frequency Consensus for Distributed Antenna Arrays With Half-Duplex Wireless Coordination. In *2020 IEEE International Symposium on Antennas and Propagation and North American Radio Science Meeting*, pages 1585–1586, July 2020. ISSN: 1947-1491.

[39] Hassna Ouassal, Ming Yan, and Jeffrey A. Nanzer. Decentralized Frequency Alignment for Collaborative Beamforming in Distributed Phased Arrays. *IEEE Transactions on Wireless Communications*, 20(10):6269–6281, October 2021. Conference Name: IEEE Transactions on Wireless Communications.

[40] Jeffrey A. Nanzer, Robert L. Schmid, Thomas M. Comberiate, and Jason E. Hodkin. Open-Loop Coherent Distributed Arrays. *IEEE Transactions on Microwave Theory and Techniques*, 65(5):1662–1672, May 2017. Conference Name: IEEE Transactions on Microwave Theory and Techniques.

[41] Don J Torrieri. Fundamental limitations on repeater jamming of frequency-hopping communications. *IEEE Journal on Selected areas in Communications*, 7(4):569–575, 1989.

[42] Don Torrieri. *Principles of spread-spectrum communication systems*, volume 1. Springer, 2005.

[43] Xiujuan Dou, Yujun Kuang, Liang Jiao, and Feng Yang. A novel parallel state search based synchronization scheme for frequency hopping system. In *2012 International Conference on Computational Problem-Solving (ICCP)*, pages 418–422, October 2012.

[44] Syed Naveen Altaf Ahmed, Pramod Kumar Meher, and A. P. Vinod. Fast acquisition and time synchronization of frequency hopping burst signals. In *2017 International Conference on Signals and Systems (ICSigSys)*, pages 175–179, May 2017.

[45] Li Tong, Shaoxuan Wang, Xinqiang Tang, Xun Wei, and Lidong Lin. USRP-based Frequency Hopping Signal Record and Playback System. In *2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*, pages 389–393, July 2016.

[46] J. A. Bondy and U. S. R. Murty. *Graph theory*. Springer, 2008.

[47] Hassna Ouassal, Torre Rocco, Ming Yan, and Jeffrey A. Nanzer. Decentralized Frequency Synchronization in Distributed Antenna Arrays With Quantized Frequency States and Directed Communications. *IEEE Transactions on Antennas and Propagation*, 68(7):5280–5288, July 2020. Conference Name: IEEE Transactions on Antennas and Propagation.

[48] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal of parallel and distributed computing*, 67(1):33–46, 2007.

[49] Anton Schlegel, Sean M. Ellison, and Jeffrey A. Nanzer. A microwave sensor with sub-millimeter range accuracy using spectrally sparse signals. *IEEE Microwave and Wireless Components Letters*, 30(1):120–123, 2020.

[50] Anton Schlegel, William R. Torres, and Jeffrey A. Nanzer. Joint sensing and communication via superposition of spectrally-sparse radar and bpsk signals. In *2022 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (AP-S/URSI)*, pages 531–532, 2022.

[51] Anton Schlegel and Jeffrey A. Nanzer. Experimental demonstration of joint sensing and communications using spectrally efficient high-accuracy range estimation. *IEEE Transactions on Microwave Theory and Techniques*, pages 1–8, 2024.

[52] Serge Mghabghab, Hassna Ouassal, and Jeffrey A. Nanzer. Wireless Frequency Synchronization for Coherent Distributed Antenna Arrays. In *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, pages 1575–1576, July 2019. ISSN: 1947-1491.

[53] Omid Abari, Hariharan Rahul, Dina Katabi, and Mondira Pant. Airshare: Distributed coherent transmission made seamless. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1742–1750, 2015.

[54] H. Nyquist. Regeneration theory. *The Bell System Technical Journal*, 11(1):126–147, 1932.

[55] Linh Manh Hoang, J. Andrew Zhang, Diep N. Nguyen, and Dinh Thai Hoang. Frequency Hopping Joint Radar-Communications With Hybrid Sub-Pulse Frequency and Duration Modulation. *IEEE Wireless Communications Letters*, 11(11):2300–2304, November 2022. Conference Name: IEEE Wireless Communications Letters.

[56] Hanjing Li, Yuwen Wang, Qiang Shu, Yongwei Li, and Yingzhu Chen. Research on DOA Estimation of Frequency Hopping Signal. In *2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC)*, pages 1339–1343, December 2017.

[57] Yuncheng Yang, Xueli Sun, and Zhaogen Zhong. A Parameter Estimation Algorithm for Frequency-Hopping Signals with a Stable Noise. In *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 1898–1904, October 2018. ISSN: 2381-0947.

[58] Jason M. Merlo and Jeffrey A. Nanzer. High Accuracy Wireless Time Synchronization for Distributed Antenna Arrays. In *2022 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (AP-S/URSI)*, pages 1966–1967, July 2022. ISSN: 1947-1491.

[59] Jason M. Merlo, Serge R. Mghabghab, and Jeffrey A. Nanzer. Wireless Picosecond Time Synchronization for Distributed Antenna Arrays. *IEEE Transactions on Microwave Theory and Techniques*, 71(4):1720–1731, April 2023. Conference Name: IEEE Transactions on Microwave Theory and Techniques.

[60] Ahona Bhattacharyya and Jeffrey A. Nanzer. Multiobjective distributed beamforming using high-accuracy synchronization and localization. *IEEE Transactions on Microwave Theory and Techniques*, 73(4):2404–2413, 2025.

[61] R. McAulay and E. Hofstetter. Barankin bounds on parameter estimation. *IEEE Transactions on Information Theory*, 17(6):669–676, 1971.

[62] J. Ziv and M. Zakai. Some lower bounds on signal parameter estimation. *IEEE Transactions on Information Theory*, 15(3):386–391, 1969.

[63] Harry L Van Trees. *Detection, estimation, and modulation theory, part I: detection, estimation, and linear modulation theory*. John Wiley & Sons, 2004.

[64] Steven M Kay. *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc., 1993.

[65] Mark A Richards. *Fundamentals of radar signal processing*. McGraw-Hill Education, 2022.

[66] S. Kay. A fast and accurate single frequency estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1987–1990, December 1989. Conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing.

[67] P. Stoica, R.L. Moses, B. Friedlander, and T. Soderstrom. Maximum likelihood estimation of the parameters of multiple sinusoids from noisy measurements. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):378–392, 1989.

[68] Pushpendra Singh, Amit Singhal, and Shiv Dutt Joshi. An efficient ml frequency estimation of a sinusoid using the secant method. In *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–5, 2017.

[69] Carl De Boor and Carl De Boor. *A practical guide to splines*, volume 27. springer New York, 1978.

[70] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[71] Ahmet Serbes and Khalid Qaraqe. A fast method for estimating frequencies of multiple sinusoidals. *IEEE Signal Processing Letters*, 27:386–390, 2020.

[72] Monson H Hayes. *Statistical digital signal processing and modeling*. John Wiley & Sons, 1996.

[73] R. Roy and T. Kailath. Esprit-estimation of signal parameters via rotational invariance techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):984–995, 1989.

[74] R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986.

[75] Mohammed Rashid and Jeffrey A. Nanzer. Frequency and Phase Synchronization in Distributed Antenna Arrays Based on Consensus Averaging and Kalman Filtering. *IEEE Transactions on Wireless Communications*, pages 1–1, 2022. Conference Name: IEEE Transactions on Wireless Communications.

[76] Mohammed Rashid and Jeffrey A. Nanzer. Frequency and phase synchronization in distributed antenna arrays based on consensus averaging and kalman filtering. *IEEE Transactions on Wireless Communications*, 22(4):2789–2803, 2023.

[77] Mohammed Rashid and Jeffrey A. Nanzer. High accuracy distributed kalman filtering for synchronizing frequency and phase in distributed phased arrays. *IEEE Signal Processing Letters*, 30:688–692, 2023.

[78] Mohammed Rashid, William R. Torres, Tammy Chang, and Jeffrey A. Nanzer. A two-tone frequency hopping waveform for decentralized consensus-based joint frequency and phase alignment in distributed antenna arrays. *IEEE Sensors Letters*, 8(3):1–4, 2024.

[79] a National Instruments Brand Ettus Research. USRP X310 High Performance Software Defined Radio - Ettus Research — ettus.com. https://www.ettus.com/all-products/x310-kit/. [Accessed 03-05-2024].

[80] a National Instruments Brand Ettus Research. UBX 10 MHz - 6 GHz Rx/Tx, 160 MHz BW - Ettus Research — ettus.com. https://www.ettus.com/all-products/ubx160/. [Accessed 03-05-2024].

[81] Jason M. Merlo and Jeffrey A. Nanzer. Wireless time and phase alignment for wideband beamforming in distributed phased arrays. In *2023 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (USNC-URSI)*, pages 365–366, 2023.

APPENDIX

```
# %% [markdown]
# # Estimation Techniques Vs. Residual Frequencies
# ##### Description: Compare the performance of estimations
techniques of each tone Vs.
# the resulting residual BW after convergence, this techniques
include coarse peak search, QLS(Praguer)
# ##### Author: William R. Torres
# ##### Last Day Modified: date: 07/08/24

# %% [markdown]
# ### Assumptions made for the simulation
# - All signals arrive at the same time and there is no window
slipping from time misalignment

# %% [markdown]
#
# ## Import all libraries and necessary dependencies

# %%
import numpy as np
import random; random.seed(0)
import matplotlib.pyplot as plt
from scipy.signal import find_peaks
from scipy.interpolate import CubicSpline
import scipy.signal as signal
from scipy.fft import fft
from scipy.fft import fftshift
from scipy.interpolate import CubicSpline
from copy import deepcopy
import math
import time
import networkx as nx

"""
Call MATLAB API to have access to the metropolis hasting network
generator provided
"""

import matlab.engine
eng = matlab.engine.start_matlab()
hard_code_exponent_for_base2 = 17

# Start timing the duration of the simulation for performance
comparison purpose
start_time = time.time()
```

```
#---
#---
# Auxiliary Functions
#---
#---
def networkPlotter(adjacency_matrix):
    rows, cols = adjacency_matrix.shape
    G = nx.DiGraph()
    for i in range(rows):
        for j in range(cols):
            if adjacency_matrix[i][j] != 0:
                G.add_edge(i,j)
    #nx.draw(G)
    return G

def base2_size(array_length):
    non_incremented_base = math.log2(array_length)
    rounded_base = math.ceil(non_incremented_base)+4
    new_length = 2**(rounded_base)
    return new_length

def bandwidth_estimator(tones_estimates):
    sorted_values = np.sort(tones_estimates)
    bw_estimates = sorted_values[1::2] - sorted_values[0::2]
    return bw_estimates
def tone_value_generator(fs=200e6, number_of_nodes=4, bw =
None):
    """
    Input(s):
    fs(float) - Sampling frequency of the system;
    number_of_nodes(int) - number of nodes/SDRs used in the
simulation
    bw(list/array or blank) - ordered list of the bandwidth of
the two-tone waveform signals
        *** If bw is not provided (it is the first iteration of
the algorithm) or the length of the bw is
            not the same as number_of_nodes
            it will generate the values to match properly :)

    return a list containig the tone's value within the
instantaneous bandwidth
    """

    # Respect the nyquist rate
    max_freq_dom_val = fs/2
    # Calculate the max bandwidth of each channel (assume that
they are the save size)
```

```
    single_channel_bw = (max_freq_dom_val - 0)/number_of_nodes
    # Determine the center point of a channels bandwidths,
distance from left boundary
    channel_bw_half_point = single_channel_bw /2
    # Calculate the value of the center of each channel, which
also corresponds to the center of the two-tone's frequencies
    central_freq = ((np.arange(start=0, stop=number_of_nodes,
step=1, dtype=int))* single_channel_bw)+ channel_bw_half_point

    ## two-tone waveforms Bandwidth generator
    # if input is not give, or the number of nodes does not
correspond to the size of the bw list
    if bw is None or len(bw) != number_of_nodes:
        # Scale the channel bandwidth to prevent overlap and
generate a bandguard, and preserve orthogonality
        two_tone_bw_range = channel_bw_half_point *
np.array([0.2,0.8])
        # Generate the two-tone BWs values
        bw = np.random.uniform(low=two_tone_bw_range[0],
high=two_tone_bw_range[1], size=(number_of_nodes,))
    # return a list of sorted values
    return bw, central_freq

def zero_pad_truncation(raw_rx_time_capture):
    ## Determine the threshold to detect
    only_positive_vals = np.absolute(raw_rx_time_capture)
    average_noise_mag = np. mean(only_positive_vals[1:200])
    threshold_val = (np.max(only_positive_vals) -
average_noise_mag)/2
    # Find the index of the first non-zero element
    first_nonzero_index = np.argmax(only_positive_vals >=
threshold_val)
    last_nonzero_index = len(only_positive_vals) -
np.argmax(np.flip(only_positive_vals) >= threshold_val) - 1
    truncated_time_array =
raw_rx_time_capture[first_nonzero_index+1:last_nonzero_index-1]
    return truncated_time_array

def coarse_peak_finder(arr_of_mag, num_of_peaks):

    first_peak_index,_= find_peaks(arr_of_mag/
np.max(arr_of_mag), height=0.001, distance=10)
    highest_peak_indices =
np.argsort(arr_of_mag[first_peak_index])[-num_of_peaks:]
    coarse_peaks_idx = first_peak_index[highest_peak_indices]
    coarse_peaks_idx = np.sort(coarse_peaks_idx)
    return coarse_peaks_idx
```

```
def praguer_qls_frequency_estimate(array_of_mag, num_of_peaks,
F, DF):

    windowing_center_idx = coarse_peak_finder(array_of_mag,
num_of_peaks)
    freq_estimates = np.zeros(num_of_peaks,dtype=float)

    d1 = coarse_peak_finder(array_of_mag, num_of_peaks)

    y0 = np.log10(array_of_mag[np.array(d1-1)])
    y1 = np.log10(array_of_mag[np.array(d1)])
    y2 = np.log10(array_of_mag[np.array(d1+1)])

    partial_idx = 1*((y2 - y0)/(2*y0 - 4*y1 + 2*y2))

    full_F = F[d1]
    partial_F = partial_idx*DF

    freq_estimate = full_F - partial_F
    return freq_estimate
def cubic_spline_frequency_estimate(array_of_mag, num_of_peaks,
F, DF):

    windowing_center_idx = coarse_peak_finder(array_of_mag,
num_of_peaks)
    freq_estimates = np.zeros(num_of_peaks,dtype=float)

    for peak in range(0, num_of_peaks):

        central_idx = windowing_center_idx[peak]

        lower_bound = central_idx - 3
        upper_bound = central_idx + 3+1
        idx_range = np.arange(lower_bound, upper_bound)

        cubicSplineFunction = CubicSpline(F[idx_range],
array_of_mag[idx_range])

        F_interp = np.linspace(F[lower_bound+0],
F[upper_bound-0], num=150)
        y_interp = cubicSplineFunction(F_interp)

        peak_index,_= find_peaks(y_interp/np.max(y_interp))
        highest_peak_indices = np.argsort(y_interp[peak_index])
[-1:]
```

```
        freq_estimates[peak] = F_interp[highest_peak_indices]
    return freq_estimates

def tx_ttwvfrm_generation(bw=500e3,fs=200e6, tt_center=0,
pulse_duration=50e-6):
    L = int(round(fs*pulse_duration))
    pulse_time_axis = np.arange(0, L)*1/fs
    pulse = np.zeros(L, dtype=complex)
    tones = np.array([-bw/2, bw/2])+tt_center
    phase = random.uniform(0, 2*np.pi)
    for tone in tones:
        pulse += (1/
np.size(tones))*np.exp(1j*2*np.pi*pulse_time_axis*tone + phase)
    return pulse

def band_limited_noise(min_freq: float, max_freq: float,
samples=1024, samplerate=1):
    """Generate band-limited white noise using Fourier method.

    Based on: https://stackoverflow.com/a/35091295/2831057

    Parameters
    ----------
    min_freq : float
        Minimum noise frequency (Hz)
    max_freq : float
        Maximum noise frequency (Hz)
        samples:
    samples : int, optional
        Lengh of vector to return (samples), by default 1024
    samplerate : int, optional
        Sample rate of noise vector (Hz), by default 1

    Returns
    -------
    array_like
        Vector or noise with properties matching specified
parameters.
    """

    def fftnoise(f):
        f = np.array(f, dtype="complex")
        Np = (len(f) - 1) // 2
        phases = np.random.rand(Np) * 2 * np.pi
        phases = np.cos(phases) + 1j * np.sin(phases)
        f[1 : Np + 1] *= phases
```

```
        f[-1 : -1 - Np : -1] = np.conj(f[1 : Np + 1])
        return np.fft.ifft(f, norm="ortho").real
    freqs = np.abs(np.fft.fftfreq(samples, 1 / fs))
    f = np.zeros(samples)
    idx = np.where(np.logical_and(freqs >= min_freq, freqs <=
max_freq))[0]
    f[idx] = 1
    noise = fftnoise(f)
    return noise

def add_complex_noise(signal, fs=200e6, snr_db=30):
    L = signal.size
    snr = 10**(snr_db / 10)
    signal_power = np.var(signal)
    #print(f'Signal Power: {10 * np.log10(signal_power)} dBm')
    noise = np.sqrt(signal_power / snr / 2) *
band_limited_noise(0, fs/2, L, fs) + 1.0j *
band_limited_noise(0, fs/2, L, fs)
    #print(f'Noise Power: {10 * np.log10(np.var(noise))} dBm')
    #print(f'SNR: {10 * np.log10(signal_power/np.var(noise))}
dB')
    rx = signal + noise

    return rx
#---
#---
#---
# Actual Simulatin Starts Here
#---
#---
#---

# Monte Carlos Simulation Parameters and memory holders
max_MC = 1000
time_range = np.logspace(-5,0, endpoint=True)

max_average_consensus_count = 50
average_consensus_count =
np.arange(0,max_average_consensus_count)

# %% [markdown]
# ## Network Generation

# %%
# the network will have this number of nodes(SDRs)
numberOfNodes = 5
# generate the network of nodes undirected communication
```

```
channels
W = np.array(eng.generateW(numberOfNodes, 0.5))
W_mixing = W.copy()
transmit_pulse_duration = 80e-6

# %%
# Get memory space for values holding
holder_bws_for_experiment =
np.zeros([max_average_consensus_count+1, numberOfNodes])
holder_for_residual = np.zeros([max_average_consensus_count+1,
1])

for k in average_consensus_count:

    ## TX Parameters
    # Sampling frequency of transmitter and receivers is going
to be constant for all nodes
    fs = 200e6 # 200MSa/s

    # length of pulse
    pulse_length = int(fs*transmit_pulse_duration)
    DT = 1/fs
    # Transmitter time axis
    tx_time_axis = np.arange(0,pulse_length,1)*DT

    ## RX Parameters
    # Capture duration
    capture_window_duration = 120e-6
    # length of capture window
    RX_window_length = int(fs*capture_window_duration)
    # Receiver time axis
    rx_time_axis = np.arange(0, RX_window_length)*DT

    ## Tone Values Generation

    # number of tones in the spectrum
    numberOfTones = numberOfNodes*2 # two-tone waveform

    if k==0:
        vals = tone_value_generator(fs=fs,
number_of_nodes=numberOfNodes, )
        holder_bws_for_experiment[k,:] = vals[0]
        holder_for_residual[k] = np.max(vals[0]) -
np.min(vals[0])
    else:
        vals = tone_value_generator(fs=fs,
number_of_nodes=numberOfNodes, bw = next_nominal_bw_holder)
```

```
    # list of two-tone bandwidths
    tt_bw = vals[0]
    # list of central frequencies axis corresponding 1-to-1
mapping with the bandwidths
    central_freq = vals[1]

    ### Generate the noiseless case for the transmitted waves
    pulse_holder = np.zeros([pulse_length,numberOfNodes],
dtype=np.complex_)
    # for every node in the system ...
    for node in range(0,numberOfNodes):
        # create a label to identify
        #identifier_label = "Node"+str(node+1) + " TX"

        # Generate and store the noiseless wave forms
        pulse_holder[:,node] =
tx_ttwvfrm_generation(tt_bw[node],fs,
tt_center=central_freq[node],

pulse_duration=transmit_pulse_duration)
        ##
==========================================================
==========================================

    ## System wide holder
    individual_node_rx =
np.zeros([numberOfNodes,RX_window_length], dtype=complex)

    # for each node ...
    for node_rx in range(0,numberOfNodes):
        # make a copy of ideal sinusodials for the node
        list_of_pulses_for_node = deepcopy(pulse_holder)

        # allocate memory for reception, (capture window)
        zero_padded_holder = np.zeros([RX_window_length],
dtype=complex)

        # for each two-tone signal ...
        zero_padding_length_one_way = int((RX_window_length-
pulse_length)/2)
        for node in range(0,numberOfNodes):
        # add noise to the channel
            noisy_pulse =
add_complex_noise(list_of_pulses_for_node[:,node]/
(np.max(list_of_pulses_for_node[:,node])),
                                              fs=fs, snr_db=30)
```

```
            # append pre/pos zero pad
            zero_padded_holder[zero_padding_length_one_way:-
zero_padding_length_one_way] = noisy_pulse
            individual_node_rx[node_rx] += zero_padded_holder
    ##
================================================================
=========================================

    # Estimation Starts From Here
    #---
    #---
    #---
    coarse_estimates = np.zeros([numberOfNodes, numberOfTones])
    coarse_bw_estimates = np.zeros([numberOfNodes,
numberOfNodes])
    next_nominal_bw_holder = np.zeros((numberOfNodes,1))
    for receiving_node in range(0,numberOfNodes):
        #plt.subplot(int(np.sqrt(numberOfNodes)),
int(np.sqrt(numberOfNodes)), receiving_node+1)
        identifier_label = "Node"+str(receiving_node+1) + " RX"
        time_rx = individual_node_rx[receiving_node]
        just_pulse = zero_pad_truncation(time_rx)

        # Convert to the fourier domain
        new_fourier_dom_axis_size = base2_size(RX_window_length)

        fourier_magnitude = np.abs(fftshift(fft(just_pulse, n=
new_fourier_dom_axis_size)))/np.size(just_pulse)
        fourier_domain_DF = fs/new_fourier_dom_axis_size
        new_F = np.arange(-fs/2, fs/2, fourier_domain_DF)

        # Coarse Frequency Finder [FFT index]
        coarse_index = coarse_peak_finder(fourier_magnitude,
numberOfTones)
        coarse_estimates[receiving_node,:] = new_F[coarse_index]

        """
        Replace the estimation function below for each different
approach

----------------------------------------------------------------
----------------------------------------
        """
        tones_estimates =
praguer_qls_frequency_estimate(fourier_magnitude, numberOfTones,
new_F, fourier_domain_DF)
        bw_estimates = bandwidth_estimator(tones_estimates)
```

```
        """

------------------------------------------------------------------
----------------------------------------
        """

        local_average = np.dot(W_mixing[receiving_node,:],
bw_estimates)
        next_nominal_bw_holder[receiving_node] = local_average
    holder_bws_for_experiment[k+1,:] =
next_nominal_bw_holder[:,0]
    holder_for_residual[k+1,:] =
np.max(next_nominal_bw_holder[:,0]) -
np.min(next_nominal_bw_holder[:,0])


# %%
end_time = time.time()
print("The time of execution of above program is :",
      (end_time-start_time), "s")
eng.quit()
```