Fully wireless implementation of distributed beamforming on a software-defined radio platform

ABSTRACT

We describe the key ideas behind our implementation of distributed beamforming on a GNU-radio based software-defined radio platform. Distributed beamforming is a cooperative transmission scheme whereby a number of nodes in a wireless network organize themselves into a *virtual antenna array* and focus their transmission in the direction of the intended receiver, potentially achieving orders of magnitude improvements in energy efficiency. This technique has been extensively studied over the past decade and its practical feasibility has been demonstrated in multiple experimental prototypes. Our contributions in the work reported in this paper are threefold: (a) the first ever all-wireless implementation of distributed beamforming without any secondary wired channels for clock distribution or channel feedback, (b) a novel digital baseband approach to synchronization of high frequency RF signals that requires no hardware modifications, and (c) an implementation of distributed beamforming on a standard, open platform that allows easy reuse and extension. We describe the design of our system in detail, present some initial results and discuss future directions for this work.

Categories and Subject Descriptors

C.2.1 [**Network architecture and design**]: Wireless communication

General Terms

Design, Experimentation

IPSN 2012 Beijing, China

Keywords

Cooperative communication, software-defined radio, distributed beamforming

1. INTRODUCTION

In this paper, we describe the key ideas behind our recent all-digital implementation of distributed transmit beamforming on a GNU-radio [9] based software-defined radio (SDR) platform. Distributed beamforming refers to a cooperative transmission scheme whereby a number of nodes in a wireless network organize themselves into a *virtual antenna array* and cooperatively transmit a common message signal to a distant receiver. This technique is especially attractive for wireless sensor networks because it allows inexpensive nodes with simple omnidirectional antennas to collaboratively emulate a highly directional antenna and focus their transmission in the direction of the intended receiver. This potentially offers large increases in energy efficiency: an array of N nodes can achieve an N^2-fold increase in the power at a receiver compared to a single node transmitting individually; conversely each node in a N-node array can reduce its transmit power by a factor of $\frac{1}{N^2}$ and still achieve the same overall signal power at the receiver compared to a single transmitter.

It is important to note that this is not just a reduction in the *per node* transmitted power simply because there are more nodes transmitting; this is also an increase in the *energy efficiency* of the transmission: a N-node beamforming array can achieve the same received signal strength (RSS) at the receiver with as little as $\frac{1}{N}$ of the *total transmit power* required by a single node transmitting individually.

Physically this increased energy efficiency arises from the increased directivity of the transmissions; the signals from the individual transmitters combine constructively at the intended receiver and as a result a larger proportion of the transmitted power is concentrated in the direction of the intended receiver. This is illustrated in Fig. 1. This requires that the signals from the individual transmitters are all aligned in phase at the intended

receiver. This in turn requires precise control of the phase of the RF signal from each transmitter.

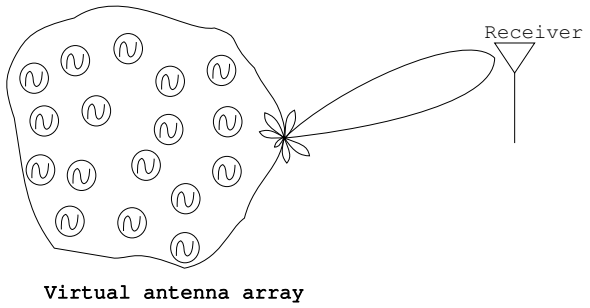

Figure 1: Energy efficient transmission using distributed beamforming.

The key challenge in realizing the large potential gains from beamforming is in precisely synchronizing the RF signals. Each transmitter in general obtains its RF carrier signal from its own local oscillator, and even when two oscillators are set to the same nominal frequency, because of manufacturing tolerances and temperature variations, they would in general have a non-zero frequency offset with respect to each other. In addition all oscillators undergo random unpredictable phase drifts over time. Finally, unlike a traditional phased array, a virtual array made up of collaborating wireless sensor nodes does not have a regular and precisely known geometry; furthermore standard localization techniques such as GPS fall far short of the accuracy necessary to overcome this geometric uncertainty for the purposes of beamforming. Thus, distributed beamforming requires a highly sophisticated synchronization process that accounts for all of the above uncertainties.

The goals of our implementation are two-fold: (a) to provide a platform for prototyping and testing algorithms for distributed beamforming and other advanced virtual array techniques, and (b) to develop and publish an open-source implementation of the basic building blocks for RF carrier synchronization to stimulate further research into advanced networking algorithms based on distributed beamforming and application of this concept to practical wireless networks.

Note that there are other cooperative transmission schemes that unlike distributed beamforming, do *not* require precise phase alignment. This includes all relaying and multi-hopping schemes where different transmitters use orthogonal space/time/frequency channels so that their transmissions do not interfere with each other. In contrast, beamforming *depends* on transmitters interfering with each other in a carefully controlled way. Orthogonal cooperation schemes can provide diversity gains in fading channels, however, they cannot provide the energy efficiency gains achievable from beamforming.

The problem of synchronizing transmitters for distributed transmit beamforming has attracted a great deal of attention over the last decade; many techniques have been developed offering different sets of tradeoffs between simplicity, overheads associated with coordination messages between the transmitters, and overheads associated with channel feedback from the receiver.

The 1-bit feedback technique introduced in [19] offers one example of this tradeoff. This algorithm has attractive properties of robustness to noise, estimation errors, and other disturbances and it dynamically adapts to channel time-variations. The 1-bit algorithm also has the very desirable property of *scalability*: the implementation of the algorithm does not depend on the number of collaborating transmitters; nodes can join and leave the virtual array at any time and the algorithm automatically adapts without any reconfiguration.

Finally the simplicity of this algorithm makes it possible to implement it on inexpensive hardware. For these reasons, we chose this 1-bit feedback algorithm as the starting point for our first implementation of distributed beamforming on the SDR platform which forms the basis for the results reported in this paper.

1.1 Contributions

Our contributions in this paper are summarized as follows.

1. **Open-source implementation of distributed beamforming.** While distributed beamforming has been experimentally demonstrated before, our SDR implementation is noteworthy in several respects:
   - To the best of our knowledge, this is the first ever all-wireless implementation of distributed beamforming; previous experimental work in [21, 32, 33] all make use of reliable wired, secondary communication channels for channel feedback and/or to distribute a reference clock signal.
   - Our implementation does not require any RF hardware modifications and performs the necessary RF signal synchronization entirely in software.
   - Previous experimental work in [21, 32] are based on custom designed hardware and as such, they are not easily reusable and extendable. The only previous implementation of beamforming that used an open platform was [33], which however uses wired distribution of common oscillator signals to all nodes as noted above and therefore does not address the syn-

chronization problem.

2. **Digital architecture for synchronization.** Our implementation of distributed beamforming is based on a novel signal processing architecture for digitally synchronizing high-frequency RF signals. This architecture is based on the observation that even at high frequencies on the order of 1 GHz, the relative frequency and phase offsets between a pair of oscillators are usually sufficiently small and slowly varying, that they can be estimated and corrected in software on standard CPUs.
3. **Low complexity algorithms for synchronization.** We present low complexity digital techniques for several important synchronization sub-problems including (a) an algorithm based on a modified version of the classical Costas feedback loop [4] for frequency locking, (b) a general method for using a reference signal at one frequency to synthesize a synchronized signal at a different frequency, and (c) a simple frequency multiplexing scheme that allows the beamforming nodes to simultaneously receive both a reference carrier signal and channel feedback from the receiver.

Outline. The rest of the paper is organized as follows. Section 2 presents some background information including a survey of previous work related to cooperative transmission techniques for wireless sensor networks. We introduce our digital architecture for synchronization in Section 3. Section 4 introduces the setup for our implementation of distributed beamforming and describes several signal processing algorithms that serve as building blocks for the implementation. We present experimental results from our implementation in Section 5 and conclude in Section 6.

2. BACKGROUND

We now present some background information and a brief survey of related work.

2.1 Cooperative transmission techniques

The large gains achievable through collaborative transmission schemes has been known to information theorists for many decades. Indeed the idea of cooperative beamforming is implicit in many early information theoretic works on multi-user channels [6]. The idea of distributed beamforming can also be further generalized to distributed MIMO [37], where nodes in a wireless network organize themselves into virtual arrays that use MIMO techniques such as spatial multiplexing and precoding to potentially achieve substantially better spatial reuse in addition to energy efficiency. In fact, it has been shown recently [24] that wireless networks using distributed MIMO can effectively overcome the famous capacity scaling limits of wireless networks due to Gupta and Kumar [10]. This literature has, however, largely ignored the synchronization requirements for achieving these cooperation gains.

More recently the concept of *user cooperation diversity* where nearby users in a cellular system use cooperation to achieve decreased outage probability in the uplink was first suggested in [31] and further developed using space-time coding theory [15, 7]. As noted earlier, cooperative diversity techniques have less stringent synchronization requirements [16] as compared to beamforming, but do not deliver the energy efficiency gains achievable with beamforming.

2.2 Experimental implementations of cooperative transmission techniques

Following up on the recent interest in cooperative communication, there have been several experimental implementations to study the practical feasibility of these ideas. This body of experimental work is summarized in a recent survey article [3], and has focused largely on cooperative diversity techniques. A recent experimental study of the amplify-and-forward relaying scheme [22] on Rice University's WARP platform [29] suggested that large gains are achievable even with a simple Alamouti space-time code. A DSP-based testbed was used for a comparative study of cooperative relaying schemes in [36]. A general testbed for systematically studying different MAC and PHY cooperative schemes was reported in [14]. Implementations of cooperative relaying have also been developed [2, 38] for software-defined radio platforms very similar to the one used in our implementation.

Diversity schemes as pointed out earlier have substantially less stringent synchronization requirements than beamforming, which makes them easier to implement. However, there have also been several recent experimental studies of distributed beamforming [21, 32, 33]. All of the above implementations have been based on the 1-bit feedback algorithm.

Distributed beamforming is also at the heart of the Coordinated Multi-Point (CoMP) systems developed as part of the European EASY-C project [11]; these make extensive use of various capabilities of cellular network infrastructure such as (a) uninterrupted availability of GPS signals, which are used to frequency-lock local oscillators and to supply symbol-level synchronization [13], (b) uplink channels with high bandwidths and low latencies to send detailed channel state feedback from the mobiles [12], and (c) a multi-gigabit backhaul network for Basestation coordination. In contrast, our work is aimed at the very different application setting of wireless sensor networks, where we cannot depend on the availability of such a sophisticated wired infrastructure.

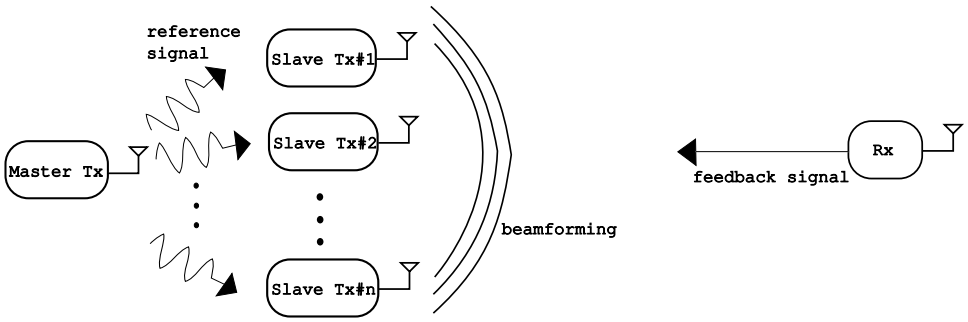

Figure 2: Experimental setup.

2.3 Synchronization techniques for distributed beamforming

While early work on cooperative communication did not focus on the synchronization issues, this changed in the last decade, and a number of synchronization techniques for distributed beamforming have now been developed (see the survey article [18]), including full-feedback closed-loop [34], 1-bit closed-loop [19, 21, 20], master-slave open-loop [17] synchronization, round-trip synchronization [5], two-way synchronization [26]. These techniques offer different sets of tradeoffs between simplicity, overheads associated with coordination messages between the transmitters, and overheads associated with feedback from the receiver.

In general, the overheads associated with the synchronization process has costs that must be weighed against the benefits available from beamforming. One of the important goals of our implementation is precisely to show that these overhead costs are modest even without expensive custom designed hardware. Specifically we used the inexpensive oscillators [23] that come standard with the radios or Universal Software Radio Peripherals (USRPs); these have frequency offsets on the order of ± 10 parts per million. In contrast, high quality ovenized oscillators with frequency tolerance of around 20 parts per *billion* are now available [27] for around 400 dollars. Highly stable chip-scale atomic clocks [35] are also now coming closer to commercial feasibility. As these high-quality oscillators become more widely used in commodity wireless hardware, the overheads associated with carrier synchronization will become correspondingly smaller and this will make cooperative techniques such as distributed beamforming even more attractive over an increasing range of frequencies.

2.3.1 *The 1-bit feedback algorithm*

The 1-bit feedback algorithm for beamforming was originally introduced in [19] and is illustrated in Fig. 3; under this algorithm, in every time-slot, each transmitter independently makes a random phase perturbation in its transmissions to the receiver; the receiver monitors the received signal strength (RSS), and broadcasts exactly 1 bit of feedback to the transmitting nodes indicating whether the RSS in the preceding time-slot was greater than in previous time-slots. Using this 1 bit of feedback, the transmitters retain the favorable phase perturbations and discard the unfavorable ones.

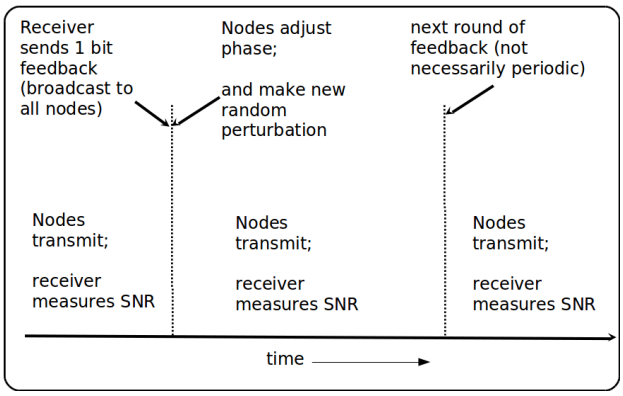

Figure 3: Illustration of the 1-bit feedback algorithm.

Over time, it can be shown [20] that the transmitters converge to coherence *almost surely* under some mild conditions on the distribution of the phase perturbations. Furthermore the algorithm is extremely robust to noise, estimation errors, lost feedback signals and time-varying phases; these attractive properties make it possible to implement this algorithm on simple hardware, and indeed as noted earlier, distributed beamforming using variations of this basic algorithm has been demonstrated on multiple experimental prototypes [21, 32, 33] at various frequencies.

Nevertheless, this algorithm and its variants suffer from a number of shortcomings.

1. *Slow convergence rate.* While the convergence rate of the 1-bit algorithm, with appropriately chosen

parameters, has good scaling properties for large arrays (convergence time increasing no faster than linearly with number of transmitters [20]), in absolute terms, it requires a large number of time-slots.

2. *Latency limitations.* The 1-bit algorithm neglects latency in the feedback channel; it assumes that the feedback signal is available instantaneously and simultaneously at all the transmitting nodes. If this assumption does not hold, maintaining time-slots across the beamforming nodes becomes much more challenging. In practice this may impose a high lower-bound on the time-slot duration which compounds the problem of slow convergence rate.
3. *Poor performance with frequency offsets.* Non-zero frequency offsets between transmitters manifest themselves as rapid time-variations in the phase. While variations of the 1-bit algorithm have been developed that can handle frequency offsets [32], these too require high feedback rates on the order of $10 \times \Delta f_{max} \times N$, where N is the number of transmitting nodes and Δf_{max} is the maximum frequency offset between the transmitters.

Recent work has shown that it is possible to overcome the above shortcomings of the 1-bit algorithm while retaining its attractive features by using richer feedback from the receiver [1]. In our experimental setup we have implemented the receiver feedback in a flexible way that allows for easy generalization to more advanced algorithms using multi-bit feedback.

The latency limitations mentioned above can be especially challenging for software-defined radio platform [30] that typically have multiple buffering stages in the data path, in addition to processing delays that depend on CPU loads and other uncontrollable factors. To get around this limitation, our current implementation uses a separate explicit mechanism for frequency locking the oscillators on the transmitters; this removes the frequency offsets and allows us to use the simple 1-bit algorithm for beamforming even with slow rates of feedback.

3. DIGITAL SYNCHRONIZATON OF HIGH-FREQUENCY RF SIGNALS

The key idea behind our implementation is that while the RF signals transmitted by the beamforming nodes are themselves not suitable for digital processing, the clock offsets between oscillators that are nominally set to the same frequency are typically quite small. For instance, even very cheap crystal oscillators [23] have worst-case frequency deviations on the order of ± 10 parts per million of the nominal center frequency. In our experimental setup, we used center frequencies around 900 MHz, and thus our clock offsets can be expected to be no greater than 9 kHz or so. In fact, our measurements with the oscillators on the USRP boards showed clock drifts that seldom exceeded 4 kHz. Furthermore, these offsets remained roughly constant over time-scales on the order of hundreds of milliseconds.

Thus, as long as we are working with *relative offsets between two oscillators*, the frequencies are small enough and their time-variations slow enough that they can be tracked and compensated in software. This is the basic rationale behind our implementation.

Different protocols for distributed beamforming have been developed that solve the above problem in ways that represent different tradeoffs between in-network coordination, feedback from the receiver and so on. For instance, under beamforming schemes using a master-slave architecture [17], there is a designated master node that supplies the reference signal $c_0(t)$, whereas under round-trip synchronization schemes [25], the receiver itself implicitly provides the reference signal. The DSP-centric architecture developed in this paper is applicable to all of these schemes.

3.1 Two synchronization sub-problems

In this paper we focus specifically on our implementation of beamforming based on the 1-bit feedback algorithm; the setup is shown in Fig. 2. Our implementation divides the beamforming problem into two subproblems.

1. **Frequency locking the transmitters.** We use a master-slave architecture to frequency-lock the transmitters. A designated "Master" node broadcasts an unmodulated tone; this tone is used as a reference signal by the "Slave" nodes to digitally correct for frequency offsets.
2. **Beamforming using 1-bit feedback.** The frequency-locking process ensures that the Slave nodes have carrier signals that are frequency-locked to each other; they still have unknown but fixed relative phase offsets. The 1-bit feedback algorithm is used to estimate and correct for these phase offsets, so the Slave nodes' transmissions are aligned in phase at the Receiver.

The role of the Master node in our setup is simply to transmit an unmodulated RF tone that the Slave nodes (digitally) lock on to. While we used a dedicated Master node in our setup for simplicity, it is straightforward to modify this setup to have the receiver itself transmit a reference tone, or to use an external reference such as the signal from a GPS satellite if it is available. Each of these alternatives have their advantages and disadvantages. Thus for instance, uninterrupted availability of a GPS synchronization signal may not be a good assumption for indoor networks or where cost and form-factor constraints preclude using dedicated GPS modules on

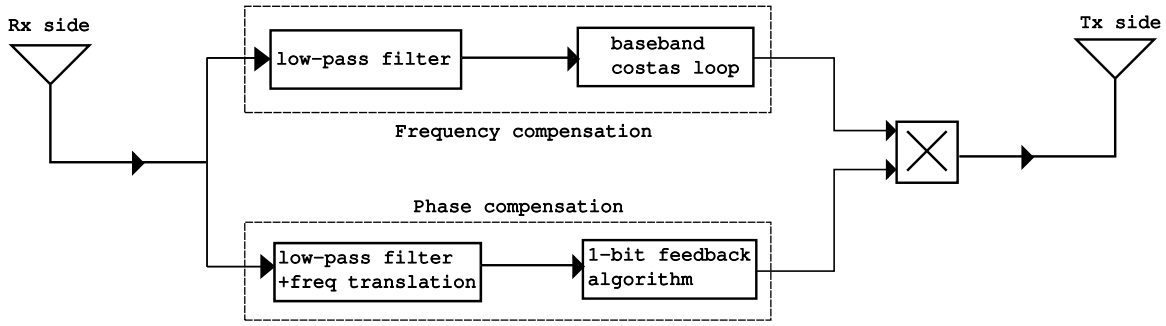

Figure 4: Signal processing at the Slave nodes.

each node. Similarly having the receiver send a reference carrier signal eliminates the need for a separate Master node, but the reference signal from a distant receiver is likely to be more noisy as compared to a signal from a Master node co-located with the Slaves.

In our setup, it is the Slave nodes that actually constitute the beamforming array, and in our implementation, most of the processing involved in synchronization and beamforming occurs at the Slave nodes. The beamforming implementation at the Slave nodes is shown in block-diagram form in Fig. 4; as indicated in the block diagram, we can think of the beamforming process at the Slave nodes as consisting of two parallel operations: frequency offset correction and phase offset correction corresponding respectively to the two steps of the synchronization process outlined above.

As noted earlier, the 1-bit feedback algorithm requires a high rate of feedback to effectively keep up with frequency offsets between transmitters. The above two-step procedure first eliminates the frequency offsets, so that the 1-bit algorithm can be effectively used with only a low rate of feedback from the distant receiver.

Before we describe our implementation of the two-step synchronization procedure, we first need to specify a frequency division multiplexing scheme for the different concurrent transmissions in this setup.

3.2 Frequency division multiplexing scheme

One important thing to note about our setup is that there are three different RF signals being transmitted by various nodes in the network simultaneously: the reference tone from the Master node to the Slaves, the beamforming signal from the Slaves towards the Receiver, and the feedback signal from the Receiver to the Slaves. Specifically, we note that the Slave nodes receive both a reference tone from the Master node and a feedback signal from the Receiver.

Thus we need to design a suitable frequency multiplexing scheme to make sure these signals do not interfere with each other, and can be extracted using relatively simple filtering operations implemented in software. In addition, we also need to ensure that *duplexing constraints* are satisfied i.e. a nodes' transmissions should not fall within the bandwidth of the same node's receiver, so there is sufficient amount of isolation between the transmit and receive hardware.

The frequency multiplexing scheme used in our experimental setup is illustrated in Fig. 5. The choice of the specific frequencies in this scheme reflects a balancing act between two conflicting objectives: one the one hand, we want to minimize the overall bandwidth of the signal received by the Slave node, so that the signal can be digitized with a relatively low sampling rate and therefore a small processing burden for the signal processing software. On the other hand, if we make the frequency separation between the reference signal from the master and the feedback signal from the receiver too small, then we will need sharp frequency-selective filters at the Slave nodes to separate the two signals, and this in turn increases the processing burden for the Slave nodes.

3.3 Simple baseband algorithm for frequency locking

We now describe the first step of the two-step synchronization process described in Section 3.1. The goal of the frequency offset correction process is to lock the RF signals transmitted by the Slave nodes to a common reference clock signal supplied by the Master node. This serves to compensate for the clock offsets between the oscillators at the Slave nodes.

Conceptually the frequency-locking problem can be formulated as follows. Given a reference signal $c_0(t) = \cos(2\pi f_1 t)$ from the Master node (i.e. a sinusoid at frequency f_1), and the pair of local oscillator signals $c_i(t) = \cos(2\pi(f_1 + \Delta f_i)t + \Delta\phi_i)$ and $s_i(t) = \sin(2\pi(f_1 + \Delta f_i)t + \Delta\phi_i)$ at Slave node i, we wish to digitally synthesize an RF signal $r_i(t) = \cos(2\pi f_2 t + \theta_i)$ at Slave i.

Note that the signals $r_i(t)$ at Slave i can have an arbi-

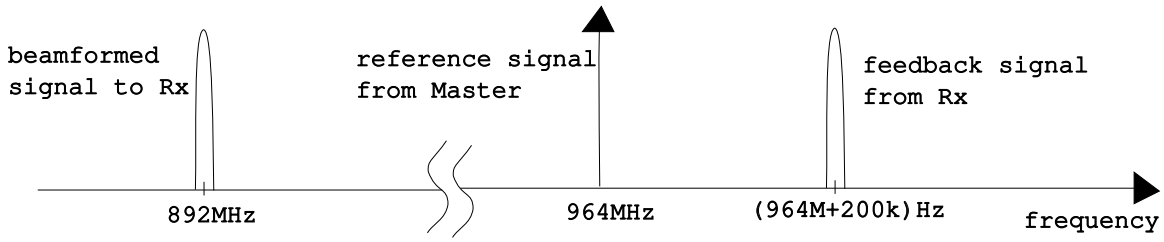

Figure 5: Frequency multiplexing scheme for beamforming experiment.

trary phase offset θ_i with each other, but must be locked to the same frequency f_2. The Slave nodes use the signals $r_i(t)$ for beamforming. As discussed in Section 3.2, because of the duplexing constraints on the Slave nodes, the beamforming frequency f_2 must be different from the frequency of the reference signal f_1. In our setup, we can see from Fig. 5 that $f_1 = 964$ MHz and $f_2 = 892$ MHz.

Our implementation achieves frequency locking by tracking the frequency *offset* between the reference signal from the Master node and the Slave's local oscillator. In our setup, we used a modified baseband version of the classic Costas loop to achieve the frequency locking; this baseband loop is shown in Fig. 7 and it works as follows.

The input to the baseband loop is the complex signal $\exp(j\phi(t))$ which represents the pair of signals $\cos\phi(t)$ and $\sin\phi(t)$, where for Slave node i, $\phi(t) = 2\pi\Delta f_i t + \Delta\phi_i$. These signals are obtained as the in-phase and quadrature components by downconverting the reference signal $c_0(t)$ using the local carrier signals $c_i(t))$ and $s_i(t)$ respectively as shown in Fig. 6.

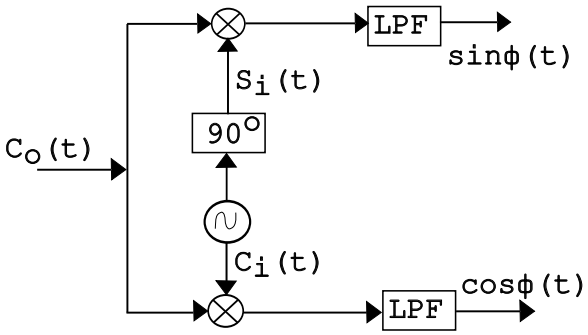

Figure 6: Oscillator offsets with reference signal.

The complex signal $\exp(j\hat{\phi}(t))$ is the output of a digital VCO with the frequency sensitivity K_1, and therefore we have by definition

$$\hat{\phi}(t) = K_1 \int_{-\infty}^{t} e(\tau)d\tau \tag{1}$$

The "error signal" $e(t)$ is obtained from the difference of $\phi(t)$ and $\hat{\phi}(t)$ as shown in Fig. 7, and this relationship can be written as

$$\begin{aligned} e(t) &= \cos\big(\phi(t) - \hat{\phi}(t)\big)\sin\big(\phi(t) - \hat{\phi}(t)\big) \\ &= \frac{1}{2}\sin\Big(2\big(\phi(t) - \hat{\phi}(t)\big)\Big) \end{aligned} \tag{2}$$

Equation (2) is mathematically equivalent to the classic Costas loop [4], though our implementation shown in Fig. 7 is quite different from the traditional RF loop. Over time, the loop makes the "error signal" $e(t)$ very small, and therefore makes $\hat{\phi}(t)$ close to $\phi(t) \equiv 2\pi\Delta f_i t + \Delta\phi_i$. In other words this baseband loop at Slave i tracks the frequency offset Δf_i between the local oscillator signal of Slave i and the reference signal $c_0(t)$.

The Slave node i is now in a position to generate frequency-locked RF signals at frequency f_1 simply by upconverting $\cos\hat{\phi}(t)$ and $\sin\hat{\phi}(t)$ using the in-phase and quadrature local oscillator signals $c_i(t)$ and $s_i(t)$ respectively. However, for beamforming, we want to generate frequency-locked carrier signals not at the same frequency f_1 as the reference signal $c_0(t)$, but rather at a *different frequency* f_2 as discussed earlier.

In order to accomplish this, we use the fact that PLL-frequency synthesizers [28] used to obtain RF signals at different frequencies can be well-modeled as frequency-multiplying devices. Thus if Slave i generates an RF carrier signal at frequency f_2 from the same underlying oscillator used to generate the signals $c_i(t)$ and $s_i(t)$ at frequency f_1, the resulting signals will have frequency offsets given by $\frac{f_2}{f_1}\Delta f_i$. In order to correct for these offsets, we need to use $\cos\hat{\phi}_2(t)$ and $\sin\hat{\phi}_2(t)$ obtained from the *scaled* offset estimate $\hat{\phi}_2(t)$ from the second VCO as shown in Fig. 7; this scaled estimate can be written as

$$\hat{\phi}_2(t) = K_2 \int_{-\infty}^{t} e(\tau)d\tau \equiv \frac{K_2}{K_1}\hat{\phi}(t) \tag{3}$$

In the above, the VCO sensitivites K_1, K_2 must be chosen to satisfy $\frac{K_2}{K_1} = \frac{f_2}{f_1}$; this ratio is equal to $\frac{892}{964}$ in our setup as shown in Fig 5.

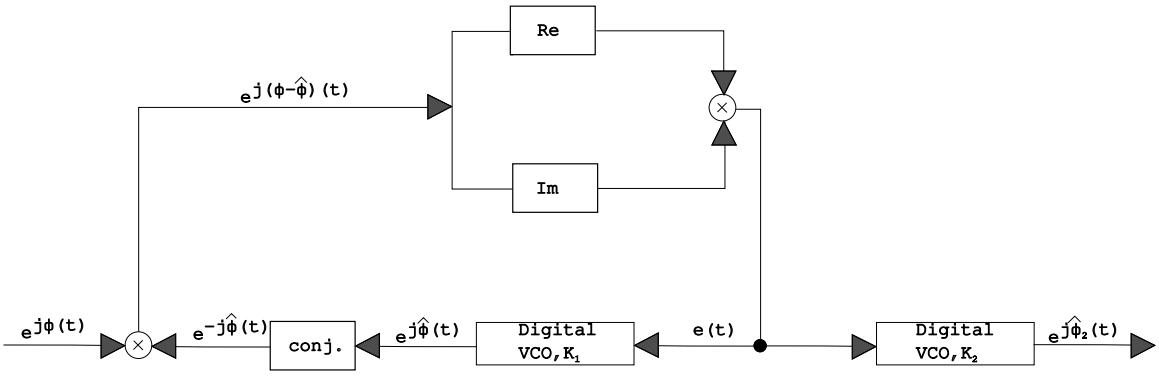

Figure 7: Modified baseband Costas loop for frequency-locking.

Note that this frequency-multiplication process may produce an unknown phase offset θ_i in the carrier signals at frequency f_2; however, this offset is constant and is easily compensated for by the 1-bit beamforming algorithm.

4. IMPLEMENTATION ON THE SOFTWARE-DEFINED RADIO PLATFORM

All the nodes used in this setup are based on the USRP RF and baseband boards [8] which is the most popular commercial SDR platform. We used the USRP-1 version of this platform, however our implementation is completely portable to the more recent versions.

Algorithm 1 Round-trip latency measurement.

Initialization:
$initial\_flag \leftarrow true$
$samp\_count \leftarrow 0$
while $initial\_flag = true$ **do**
  Average every 1000 samples to get an $RSS\,estimate$
  Compare $RSS\,estimate$ with a pre-defined $threshold$
  if $RSS\,estimate \geq threshold$ **then**
    $initial\_flag \leftarrow false$
    //Round-trip latency in number of samples:
    $r\_t\_latency \leftarrow samp\_count$
    $avg\_st\_time \leftarrow r\_t\_latency + (1mS \times samp\_rate)$
    $avg\_end\_time \leftarrow r\_t\_latency + (21mS \times samp\_rate)$
    $bf\_t\_slot\_end \leftarrow r\_t\_latency + (22mS \times samp\_rate)$
    //Round-trip latency in milli-seconds:
    $r\_t\_latency \leftarrow (samp\_count/samp\_rate) \times 1000$
  end if
end while

The 1-bit feedback algorithm requires periodic feedback of 1 bit per time-slot from the receiver regarding the received signal strength (RSS) of the beamforming signal in the previous time-slot. In our implementation, the receiver simply sends a continuous wave signal proportional to the amplitude of the received signal. This signal is broadcast wirelessly to all the beamforming nodes. This feedback signal, of course, provides a lot more than 1 bit of feedback information, and indeed we designed our feedback channel in a flexible way to permit easy generalization of our implementation to more sophisticated algorithms [1] to take advantage of richer feedback information.

Each Slave node receives this feedback signal with a delay because of latencies in the software-defined radio system; we need to first estimate the round-trip (RT) latency between each Slave and the receiver in order to extract the 1-bit feedback required for the beamforming algorithm. We described our implementation of the frequency-locking process in Section 3.3 which forms the first synchronization subproblem outlined in Section 3.1. We now describe our implementation of the second subproblem i.e. the 1-bit beamforming algorithm. The beamforming algorithm on each Slave node consists of an initialization procedure that measures the round-trip latency in the feedback channel, followed by the actual implementation of the beamforming algorithm.

The latency measurement algorithm is based on the following simple idea. Initially when none of the beamforming nodes are transmitting, the signal level at the receiver consists of just background noise which is quite small and therefore the amplitude of the feedback signal is also correspondingly small. Then when one of the Slaves starts transmitting, it can estimate its RT latency simply by counting the number of samples it takes before it sees an increase in the amplitude of the feedback signal from the receiver. This, of course, requires that each Slave node be calibrated individually. In our setup, we do this by using special flags in the software

Parameter	Variable name	Value
Round-trip latency	r_t_latency	≈30 ms
Averaging start time	avg_st_time	(r_t_latency+1)ms
Averaging end time	avg_end_time	(r_t_latency+21)ms
Beamforming time-slot end time	bf_t_slot_end	(r_t_latency+22)ms
Low-pass filter bandwidth	-	30kHz
Low-pass filter transition width	-	20kHz
Frequency correction factor of Costas loop	-	892/964
VCO sensitivity of Costas loop	-	100k rad/s/V
Baseband sampling rate	samp_rate	2 Msps
FPGA Decimation	-	32
FPGA Interpolation	-	64
Random phase perturbation distribution	-	uniform
Random phase perturbation angle	rand_pert	±15 degrees
Past RSS window size	past_rss_win	4

Table 1: Key parameters.

that can be switched on and off in real-time to start and stop transmitting from each Slave node.

The pseudo-code for the initialization process and the beamforming algorithm are given in Algorithms 1 and 2 respectively. Key parameter values along with corresponding variable names referred to in the pseudo-code are in Table 1.

5. RESULTS

We now show some experimental results from our implementation. Fig. 9 shows a photograph of the receiver node in our experimental setup which is where the measurements reported in this section were recorded. In addition to the "Flex 900" RF daughterboard that the receiver node uses for receiving the beamforming signal and for transmitting the feedback signal, we also connected an additional "Basic Tx" daughterboard to the receiver node to enable us to view the received signal strength at the receiver on an external oscilloscope. This setup is illustrated in Fig. 8.

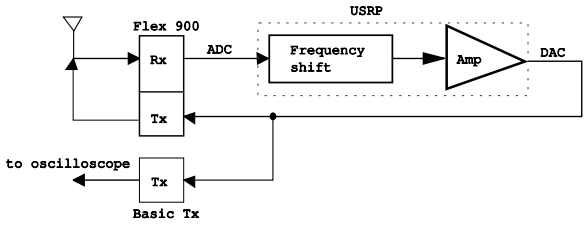

Figure 8: Measurement setup for beamforming experiment.

Figs. 10, 11 show screenshots from the oscilloscope of two runs of the beamforming experiment. Specifically, Figs 10 and 11 show the amplitude of the received signal from the beamforming Slaves, with each Slave node transmitting individually at first, and then trans-

Algorithm 2 1-bit feedback beamforming algorithm

Initialization:
$samp\_count \leftarrow 0$
$past\_rss\_win \leftarrow 0$
//*cum_phase* is the cumulative phase of a slave during a time-slot.
$cum\_phase \leftarrow 0$
while $initial\_flag = false$ **do**
  if $avg\_st\_time \leq samp\_count < avg\_end\_time$ **then**
    Average the received signal samples to obtain *current_rss*, the estimate of RSS of current time-slot
  else if $samp\_count = avg\_end\_time$ **then**
    Compare *current_rss* with *past_rss_win*
    if $current\_rss > past\_rss\_win$ **then**
      $feedback\_bit \leftarrow true$
    else
      $feedback\_bit \leftarrow false$
    end if
    From $\pm rand\_pert$, generate random phase perturbation as *c_rand_pert*
    $cum\_phase \leftarrow cum\_phase + c\_rand\_pert$
    if $feedback\_bit = false$ **then**
      $cum\_phase \leftarrow cum\_phase - p\_rand\_pert$
    end if
    Shift the FIFO *past_rss_win* by 1 to save *current_rss* in it
    Save *c_rand_pert* as *p_rand_pert*
  else if $samp\_count = bf\_t\_slot\_end$ **then**
    $samp\_count \leftarrow 0$
  end if
end while

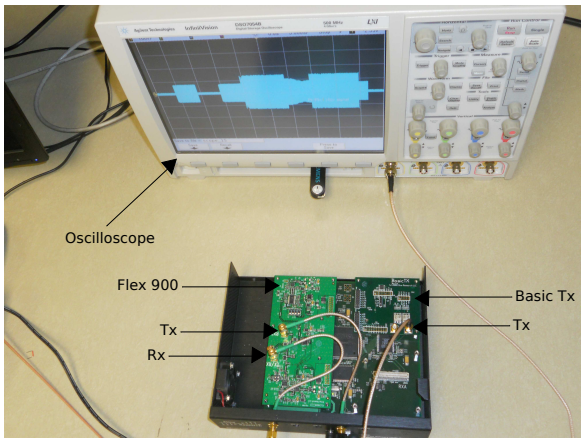

Figure 9: Photograph of measurement setup.

mitting together while implementing the beamforming algorithm and then finally transmitting together incoherently (i.e. without running the beamforming algorithm). It is also possible to dispense with the external oscilloscope completely and simply save samples of the received signal at the receiver node for offline processing and plotting; a typical result is shown in the plot in Fig. 12 which represents another run of the beamforming experiment with the same sequence of steps as Figs. 10, 11.

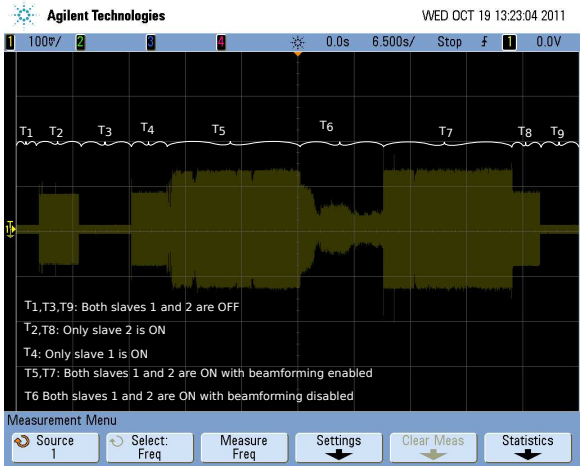

Figure 10: Received signal amplitude at the receiver - I.

The coherent gains from beamforming are apparent from the three plots. In other words, the amplitude of the received signal when the two Slaves are transmitting together is seen to be close to the sum of their individual amplitudes. It can also be seen that the beamforming gains quickly deteriorate when the two Slaves are trans-

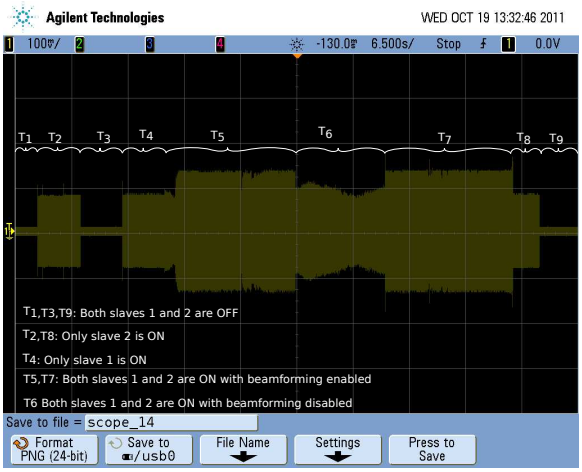

Figure 11: Received signal amplitude at the receiver - II.

mitting together but incoherently i.e. with the beamforming algorithm disabled.

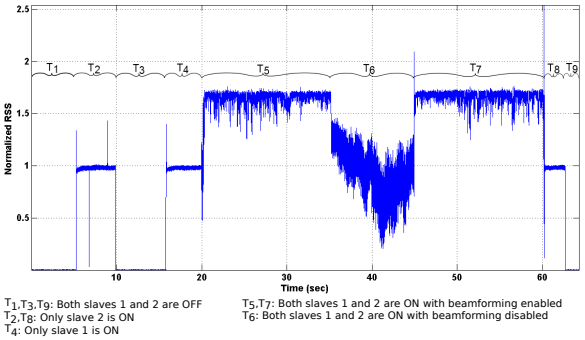

Figure 12: Received RSS at the receiver - III.

The plot in Fig. 13 also shows the "transient" of the beamforming process; specifically it shows the amplitude of the received signal, with one Slave transmitting individually at first, then the second Slave being turned on with the beamforming algorithm activated on both nodes. It is seen that the convergence time of the beamforming algorithm is on the order of several hundred milliseconds, which represents around 15 timeslots.

6. CONCLUSIONS

We described our implementation of distributed beamforming on an open software-defined radio platform. This implementation is based on a novel signal processing architecture for the synchronization of high frequency RF signals entirely in software. Our results show that the synchronization requirements for beamforming can be satisfied with modest overheads on inexpensive com-

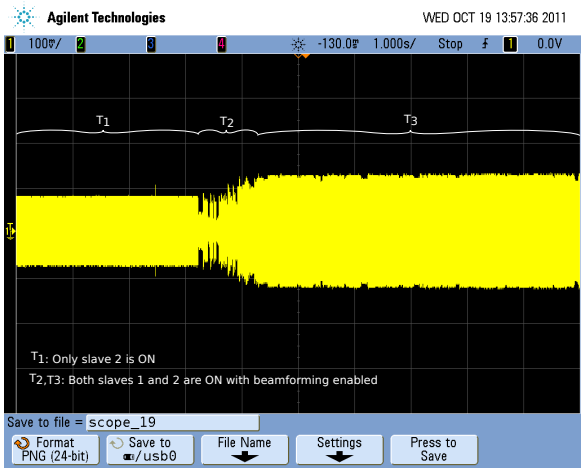

Figure 13: Rise time for beamforming - IV.

modity platforms without any hardware modifications and without any wired infrastructure. This opens up many interesting possibilities for future work in further developing open-source building blocks for bringing the large potential gains from virtual antenna arrays to real-world wireless networks. In addition, this poses a challenge of designing effective networking protocols to take advantage of cooperative communication schemes such as beamforming.

7. REFERENCES

[1] author list redacted for double-blind review. The unslotted feedback approach to distributed beamforming. In *Proceedings of the 37th International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (in review)*, 2012.

[2] G. Bradford and J. Laneman. An experimental framework for the evaluation of cooperative diversity. In *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, pages 641 –645, march 2009.

[3] G. Bradford and J. Laneman. A survey of implementation efforts and experimental design for cooperative communications. In *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 5602 –5605, march 2010.

[4] J. Costas. Synchronous communications. *Proceedings of the IRE*, 44(12):1713 –1718, dec. 1956.

[5] D.R. Brown III and H.V. Poor. Time-slotted round-trip carrier synchronization for distributed beamforming. *IEEE Trans. on Signal Processing*, 56(11):5630–5643, November 2008.

[6] A. El Gamal and T. Cover. Multiple user information theory. *Proceedings of the IEEE*, 68(12):1466 – 1483, dec. 1980.

[7] E. Erkip, A. Sendonaris, A. Stefanov, and B. Aazhang. Cooperative communication in wireless systems. In *Advances in network information theory: DIMACS Workshop Network Information Theory*, volume 66, page 303. Amer Mathematical Society, 2004.

[8] USRP products. `http://www.ettus.com/products`, 2011.

[9] Gnu radio. `http://gnuradio.org/redmine/projects/gnuradio/wiki`, 2011.

[10] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388 –404, mar 2000.

[11] R. Irmer, H. Droste, P. Marsch, M. Grieger, G. Fettweis, S. Brueck, H.-P. Mayer, L. Thiele, and V. Jungnickel. Coordinated multipoint: Concepts, performance, and field trial results. *Communications Magazine, IEEE*, 49(2):102–111, february 2011.

[12] V. Jungnickel, L. Thiele, T. Wirth, T. Haustein, S. Schiffermuller, A. Forck, S. Wahls, S. Jaeckel, S. Schubert, H. Gabler, C. Juchems, F. Luhn, R. Zavrtak, H. Droste, G. Kadel, W. Kreher, J. Mueller, W. Stoermer, and G. Wannemacher. Coordinated multipoint trials in the downlink. In *GLOBECOM Workshops, 2009 IEEE*, pages 1 –7, Dec, 2009 2009.

[13] V. Jungnickel, T. Wirth, M. Schellmann, T. Haustein, and W. Zirwas. Synchronization of cooperative base stations. In *IEEE International Symposium on Wireless Communication Systems (ISWCS)*, pages 329–334, October 2008.

[14] T. Korakis, M. Knox, E. Erkip, and S. Panwar. Cooperative network implementation using open-source platforms. *Communications Magazine, IEEE*, 47(2):134 –141, february 2009.

[15] J. Laneman and G. Wornell. Distributed space-time-coded protocols for exploiting cooperative diversity in wireless networks. *IEEE Transactions on Information Theory*, 49(10):2415 – 2425, oct. 2003.

[16] J. Mietzner, J. Eick, and P. Hoeher. On distributed space-time coding techniques for cooperative wireless networks and their sensitivity to frequency offsets. In *Smart Antennas, 2004. ITG Workshop on*, pages 114 – 121, march 2004.

[17] R. Mudumbai, G. Barriac, and U. Madhow. On the feasibility of distributed beamforming in wireless networks. *IEEE Trans. on Wireless Communication*, 6(5):1754–1763, May 2007.

[18] R. Mudumbai, D.R. Brown III, U. Madhow, and H.V. Poor. Distributed transmit beamforming: Challenges and recent progress. *IEEE Communications Magazine*, 47(2):102–110, February 2009.

[19] R. Mudumbai, J. Hespanha, U. Madhow, and G. Barriac. Scalable feedback control for distributed beamforming in sensor networks. In *IEEE International Symp. on Information Theory (ISIT)*, pages 137–141, Adelaide, Australia, September 2005.

[20] R. Mudumbai, J. Hespanha, U. Madhow, and G. Barriac. Distributed transmit beamforming using feedback control. *IEEE Trans. on Inform. Theory*, 56(1):411–426, January 2010.

[21] R. Mudumbai, B. Wild, U. Madhow, and K. Ramchandran. Distributed beamforming using 1 bit feedback: from concept to realization. In *44th Allerton Conf. on Comm., Control, and Computing*, pages 1020 – 1027, Monticello, IL, Sep. 2006.

[22] P. Murphy and A. Sabharwal. Design, implementation, and characterization of a cooperative communications system. *IEEE Transactions on Vehicular Technology*, 60(6):2534 –2544, july 2011.

[23] Datasheet for EC2620ETTS-64.000M oscillator. `http://www.ecliptek.com/SpecSheetGenerator/specific.aspx?PartNumber=EC2620ETTS-64.000M`, 2011.

[24] A. Ozgur, O. Leveque, and D. Tse. Hierarchical cooperation achieves optimal capacity scaling in ad hoc networks. *IEEE Transactions on Information Theory*, 53(10):3549–3572, Oct. 2007.

[25] I. Ozil and D.R. Brown III. Time-slotted round-trip carrier synchronization. In *Proceedings of the 41st Asilomar Conference on Signals, Systems, and Computers*, pages 1781 – 1785, Pacific Grove, CA, November 4-7, 2007.

[26] R. Preuss and D.R. Brown III. Two-way synchronization for coordinated multi-cell retrodirective downlink beamforming. *IEEE Trans. on Signal Processing*, November Accepted to appear in 2011.

[27] Rakon RFPO45 SMD oven controlled crystal oscillator datasheet. `http://www.rakon.com/Products/Public_Documents/Specifications/RFPO45.pdf`, 2009.

[28] B. Razavi. Challenges in the design of frequency synthesizers for wireless applications. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 395 –402, may 1997.

[29] Warp: Wireless open access research platform. `http://warp.rice.edu/trac`, 2011.

[30] T. Schmid, O. Sekkat, and M. B. Srivastava. An experimental study of network performance impact of increased latency in software defined radios. In *Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, WinTECH '07, pages 59–66, 2007.

[31] A. Sendonaris, E. Erkip, and B. Aazhang. Increasing uplink capacity via user cooperation diversity. In *Proc. 1998 IEEE International Symp. on Information Theory*, page 156, Cambridge, MA, August 1998.

[32] M. Seo, M. Rodwell, and U. Madhow. A feedback-based distributed phased array technique and its application to 60-ghz wireless sensor network. In *Microwave Symposium Digest, 2008 IEEE MTT-S International*, pages 683 –686, june 2008.

[33] S. Sigg and M. Beigl. Algorithms for closed-loop feedback based distributed adaptive beamforming in wireless sensor networks. In *5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 25–30. IEEE, 2009.

[34] Y. Tu and G. Pottie. Coherent cooperative transmission from multiple adjacent antennas to a distant stationary antenna through AWGN channels. In *IEEE Vehicular Technology Conf. (VTC)*, volume 1, pages 130–134, Birmingham, AL, Spring 2002.

[35] D. Youngner, L. Lust, D. Carlson, S. Lu, L. Forner, H. Chanhvongsak, and T. Stark. A manufacturable chip-scale atomic clock. In *International Solid-State Sensors, Actuators and Microsystems Conference, TRANSDUCERS*, pages 39 –44, june 2007.

[36] P. Zetterberg, C. Mavrokefalidis, A. S. Lalos, and E. Matigakis. Experimental investigation of cooperative schemes on a real-time dsp-based testbed. *EURASIP J. Wireless Comm. and Networking*, 2009.

[37] H. Zhang and H. Dai. On the capacity of distributed mimo systems. In *Proc. Conference on Information Sciences and Systems*, 2004.

[38] J. Zhang, J. Jia, Q. Zhang, and E. Lo. Implementation and evaluation of cooperative communication schemes in software-defined radio testbed. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.