Distributed Beamforming with Binary Signaling

Mark Johnson
Dept. of EECS
University of California, Berkeley
Berkeley, CA 94720, USA
mjohnson@eecs.berkeley.edu

Michael Mitzenmacher
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, 02138
michaelm@eecs.harvard.edu

Kannan Ramchandran
Dept. of EECS
University of California, Berkeley
Berkeley, CA 94720, USA
kannanr@eecs.berkeley.edu

***Abstract*— We consider a distributed beamforming problem in which nodes are restricted to sending binary phases to a receiver that has access to a one-bit feedback channel. Our simplified model allows us to prove lower bounds, as well as explore algorithmic techniques and analyses. We demonstrate both upper and lower bounds on the convergence time that are linear in the number of nodes in the system. Our upper bound is given by analyzing a simple randomized algorithm. We also discuss methods for accurately approximating the convergence time numerically that apply to our algorithm, as well as more general algorithms. Finally, we investigate modifications of the basic algorithm which improve the constant factor in the running time.**

I. Introduction

Beamforming is a technique that can be employed to significantly increase communication efficiency in sensor networks [1], [2]. By carefully synchronizing the phases of many transmitters, their signals will constructively interfere at a receiver and large SNR gains can be realized. However, achieving this synchronization in a distributed environment is a challenging problem, particularly when there are stringent cost and complexity constraints on the nodes. An algorithm that yields impressive results through the use of a single bit of feedback from the receiver to the transmitters has been recently proposed [3]. Both analysis [4], [5] and experimental prototyping [6] have confirmed its advantages.

Our work, inspired by Mudumbai, et al. [3], takes a different approach to the beamforming problem. Instead of considering nodes that can transmit carrier signals at arbitrary, continuous valued phases, we restrict the nodes to send one of two signals, denoted by $X = +1$ and $X = -1$. Similarly, the corresponding channel output is one of the same two signals. While this restriction to binary signal phases is not representative of wireless channels, it allows us to apply powerful combinatorial techniques to the problem. In particular, we are able to prove a rigorous lower bound on the convergence time of a class of distributed beamforming algorithms. We also use this simplified model to explore possible algorithmic techniques and analyses. We provide a simple randomized algorithm and give a closed-form upper bound on its performance; we also examine techniques to numerically determine its performance more exactly. We believe that the insights derived from this study of the binary problem will lead to similar results for the case with $M > 2$ transmit phases, which more accurately models real communication systems. The problem also appears interesting in its own right.

II. Problem Statement

We consider the communication system shown in Figure 1. There are N distributed transmitters which communicate with a receiver over noisy channels. In round j, node i sends a value $\hat{C}_i[j] \in \{+1, -1\}$. The output of the channel is $X_i[j] = C_i \cdot \hat{C}_i[j]$, where $C_i \in \{+1, -1\}$ denotes the channel state. The channel state vector C^N is fixed, but unknown to the transmitters. The receiver measures $Y[j] = |\sum_{i=1}^{N} X_i[j]|$, the magnitude of the received signal, and then broadcasts one bit of feedback to the transmitters over a noiseless channel. The feedback $Z[j]$ can in general be a function of $(Y[1], Y[2], \ldots, Y[j])$. Each signal $\hat{C}_i[j]$ can be based on all previous history, and a random number generator R_i. To reduce complexity, we consider algorithms that base the signal $\hat{C}_i[j+1]$ on the feedback at time j, the previous signal $\hat{C}_i[j]$, and R_i.

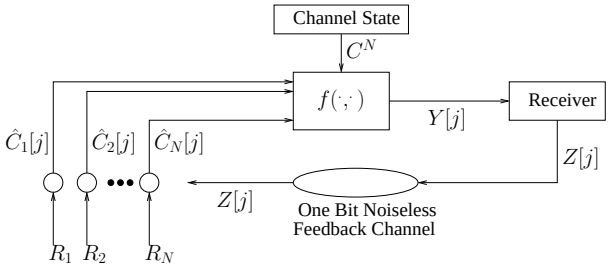

Fig. 1. Model of the communication system

The goal is for the transmitters to choose signals such that $Y[j] \geq \beta N$ for a given parameter $\beta < 1$, i.e., a constant fraction of the maximum possible signal magnitude. When $Y[j] \geq \beta N$ for the first time, we say that the algorithm has converged. Because the transmitters are spatially distributed, they cannot coordinate. Hence, trivial solutions that sequentially cycle through the transmitters are not viable.

III. A Randomized Beamforming Algorithm

We propose a simple randomized beamforming algorithm. The intuition behind our approach is to try to mimic the trivial centralized solution, which changes the transmission of exactly one node each round to check if this increases the magnitude

at the receiver. Such an algorithm could achieve $Y = N$ after only N rounds. We use randomization to make up for the lack of central coordination.

Our algorithm begins at round $j = 0$ with each node choosing its signal independently and uniformly at random, so that the initial channel outputs are $X_i[0] = 1$ or $X_i[0] = -1$, with probability $1/2$ each. Note that a node toggling its transmitted signal $\hat{C}_i$ is equivalent to toggling the channel output X_i; we use the expressions interchangeably. At each subsequent round, the algorithm proceeds as follows:

1) Each node flips an independent coin that lands on heads with probability $p = \frac{\gamma}{N}$, with γ a fixed constant that will be subsequently optimized.
2) All nodes whose coins were heads will toggle their signal X_i. The nodes whose coins were tails transmit the same signal as in the previous round.
3) The receiver will compare the value $Y[j]$ that it measures at the current iteration to $Y[j-1]$. If $Y[j] > Y[j-1]$, then the receiver sends $Z[j] = +1$ as feedback, otherwise the receiver sends $Z[j] = -1$.
4) When the receiver sends $Z[j] = +1$, the nodes that toggled their signal in the current round change their signal to its new value, and the algorithm then returns to step 1. When the receiver sends $Z[j] = -1$, the nodes that toggled their signal instead reject the change and revert to their previous value.

The operation of the algorithm is quite intuitive: in each round, a random set of nodes flip their signals, and if this new configuration results in an increase in Y, then the new configuration is retained. Otherwise, the new configuration is disregarded. It follows that $Y[j]$ is a non-decreasing function of j. Further, because of the symmetry of the system, $Y[j]$ is a Markov chain. That is, it does not matter which transmitters send the same value, or whether that value is $+1$ or -1. All that matters is the number of transmitters whose output agrees.

A. Mean Convergence Time Analysis

We prove that this algorithm must converge in expected time $O(N)$ via a simple upper bound approach. As stated above, $Y[j]$ defines the state of a Markov chain. For the purpose of simplifying the analysis of the convergence time, we consider a modified Markov chain, in which the only valid transitions are those which occur due to exactly one node toggling its signal. In the modified chain, whenever more than one node toggles its signal in a given round, the new configuration is automatically rejected. A simple coupling argument clarifies that the convergence time of the modified chain is still an upper bound on the convergence time of the true chain.

The number of rounds until the modified chain first reaches the state βN obviously depends on the initial state of the process at time $j = 0$. Because we are interested in bounding the convergence time, we analyze the worst case where the modified chain starts in state 0, that is $Y[0] = 0$. Note that Y increases by two when a successful change occurs. For convenience, we let $Y'[j] = (Y[j] + N)/2$, so that when Y jumps by two, Y' jumps by one. In words, Y' is the number of transmitters sending the "right" signal; that is, the number whose output is the same as the sign of $\sum_{i=1}^{N} X_i[j]$. In this setting, the convergence condition requires $Y'[j]$ to reach $\frac{1+\beta}{2}N = \alpha N$. In what follows, when we refer to the state, we mean the value of $Y'[j]$.

We define p_i as the probability of Y' transitioning from state i to state $i+1$ in the modified chain. When $i = N/2$, p_i is simply the probability that one node toggles its value. When $i > N/2$, we observe that p_i is equal to the probability that exactly one node toggles multiplied by the probability that this node had been transmitting the "wrong" signal (with output opposite to the sign of $\sum_{i=1}^{N} X_i[j]$). The probability that exactly one node toggles is equal to

$$\binom{N}{1} \cdot \frac{\gamma}{N} \cdot \left(1 - \frac{\gamma}{N}\right)^{N-1} = \gamma \left(1 - \frac{\gamma}{N}\right)^{N-1}$$

which approaches $\gamma e^{-\gamma}$ as N gets large. We use this approximation freely henceforth, as it only slightly affects constant factors in the asymptotic analysis. Given that exactly one node toggles its transmitted signal, that node is equally likely to be any of the N nodes in the system, so the probability that the change increases the received signal Y is $\frac{N-Y'[j]}{N}$. Therefore, $p_i = \frac{\gamma e^{-\gamma}(N-i)}{N}$ for $i > N/2$. As we are only seeking an upper bound on the convergence time, we may also use this formula to compute p_i when $i = N/2$.

The number of rounds T until the modified chain moves from state $N/2$ to state αN is the sum of a sequence of geometric random variables T_i with parameters p_i That is,

$$T = \sum_{i=N/2}^{\alpha N - 1} T_i.$$

The expectation of T is computed as

$$\begin{aligned} E[T] &= \sum_{i=N/2}^{\alpha N-1} E[T_i] \\ &= \sum_{i=N/2}^{\alpha N-1} \frac{1}{p_i} \\ &= \sum_{i=N/2}^{\alpha N-1} \frac{1}{\gamma e^{-\gamma}} \frac{N}{N-i} \\ &\approx \frac{Ne^{\gamma}}{\gamma} \ln\left(\frac{1}{2(1-\alpha)}\right). \end{aligned}$$

Thus, the average time required to satisfy the convergence condition $E[T]$ is in fact linear in N (regardless of the choice of γ). For the modified chain, the optimal value of γ is 1, giving an upper bound (up to lower order terms) of

$$E[T] \approx Ne \ln\left(\frac{1}{1-\beta}\right).$$

It is worth noting that the asymptotic behavior in $1 - \beta$ found in this upper bound is to be expected. Any algorithm that is based on a constant-sized subset of transmitters chosen uniformly at random attempting to change their behavior will

face bounds implicit from the coupon collector's problem. In particular, when $\beta = 1$, the expected time would be $\Omega(N \ln N)$.

IV. Converse

The upper bound naturally raises the question of what the minimum expected convergence time is, and in particular whether our algorithm is order optimal. In this section, we prove a corresponding $\Omega(N)$ lower bound.

First, we note that if the Hamming distance d_H between the channel state vector C^N and the transmitted signals $\hat{C}^N$ equals DN, then $Y = |(1-2D)N|$. If $Y \geq \beta N$, then either $D \leq \frac{1-\beta}{2}$ or $D \geq \frac{1+\beta}{2}$, i.e., there must be enough nodes sending $+1$ or enough sending -1 to reach the target received magnitude.

We achieve our lower bound via a reduction, by relaxing the binary beamforming problem and then giving a correspondence between any algorithm for this new problem and a standard point-to-point communication problem. We begin by modifying the actual system in Figure 1 to produce the virtual system shown in Figure 2, where the virtual encoder contains the noisy channels and the receiver while the virtual decoder contains the N transmitters. Note that the common randomness R is independent of the source C^N.

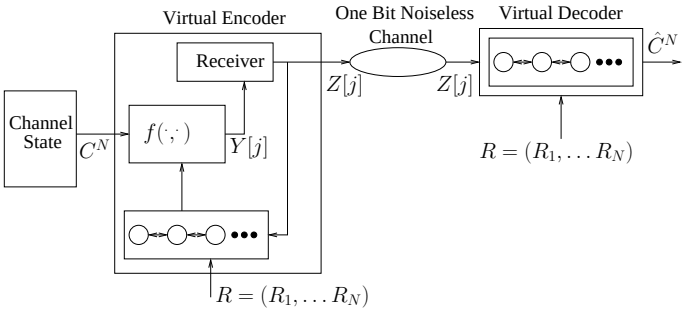

Fig. 2. The virtual communication system derived from the actual system.

There are differences between the virtual system and the actual system, which all give the virtual encoder more information and freedom, and therefore can only reduce the number of channel uses required to achieve the convergence condition:

- The N nodes can be globally coordinated.
- The virtual encoder has complete access to the random bits R. Because of this, the encoder can exactly replicate the behavior of the decoder without feedback.
- The virtual encoder output $Z[j]$ can be an arbitrary function of C^N and R, instead of being constrained to be computed by our beamforming algorithm.

The virtual system is a special case of a general point-to-point communication system. From Shannon's source-channel separation theorem [7] and from the rate-distortion bound for Bernoulli($\frac{1}{2}$) sources under a Hamming distortion constraint, it is clear that with high probability, for any ϵ and for large enough N, at least $N(R(D)-\epsilon) = N(1-h(D)-\epsilon)$ channel uses are required to obtain a reconstruction with distortion $d_H(C^N, \hat{C}^N) \leq DN$. Furthermore, if the goal is to obtain a reconstruction with distortion either less than DN or greater than $(1-D)N$, then at least $N(1-h(D)-\epsilon)-1$ channel uses are required. (This follows since with 1 additional bit the encoder can instruct the decoder whether or not to invert its current reconstruction, to obtain a distortion less than DN with $N(1-h(D)-\epsilon)$ channel uses.) Because our virtual system is a special case of a general point-to-point system, and the virtual system is strictly more informed than the real system, the convergence time of the beamforming algorithm cannot be less than $N(1-h(D)-\epsilon)-1$.

This lower bound is in fact quite general. For example, it can be extended to show that a linear number of rounds would be required to reach the convergence condition if the receiver could send a constant number of feedback bits (greater than 1, but independent of N) per round, or if the initial transmitted signals were chosen independently and identically but with a bias.

V. Approximations of Convergence Time

While these upper and lower bounds prove that the convergence time is linear in N, they are both quite loose. For example, for $N = 1000$ and $\beta = 0.7$, the upper bound is equal to 3273 and the lower bound is equal to 119. We therefore also develop numerical methods to approximate the expected convergence time of our algorithm, both asymptotically and for finite values of N.

To begin, we note that the original Markov chain is necessarily more complex than the modified chain used to derive the closed-form upper bound, as larger jumps can occur and jumps occur in many different ways. For example, one way for Y' to increase by one is if three transmitters decide to toggle their signal, and two of them are currently wrong. This logic extends to further cases. Let $f_i = (N-i)/N$, and let $p_{i,j}$ be the probability of Y' transitioning from state i to state j in the original chain. To simplify, we think of N going to infinity, and use the Poisson approximation that k transmitters attempt to toggle their value with probability $e^{-\gamma}\gamma^k/k!$. With this in mind, we find for $i > N/2$,

$$p_{i,i+j} = \sum_{k=0}^{\infty} \frac{e^{-\gamma}\gamma^{2k+j}}{(k+j)!k!}(1-f_i)^{k+j}(f_i)^k. \qquad (1)$$

Unlike for our simplified chain, we have not derived a simple closed form expression for the expected convergence time for this chain. We make one high-level point, verified by analysis and experiment. For reasonable values of β we find it better to choose $\gamma > 1$; this lessens the chance of having only one transmitter toggle, but the additional chances for gain from two (or more) transmitters toggling makes up for this. Indeed, the best choice would be to have the parameter γ vary with time, but this is beyond the scope of this summary.

Equation 1 can be used to derive fluid limit (mean field) equations that allow numerical approximations to be derived. Specifically, in the limiting system, if f is the fraction of

transmitters with the correct sign, let

$$p_j = \sum_{k=0}^{\infty} \frac{e^{-\gamma}\gamma^{2k+j}}{(k+j)!k!}(1-f)^{k+j}(f)^k.$$

Then, after scaling time so that N rounds occur per unit time, the corresponding mean field differential equation is

$$\frac{df}{dt} = \sum_{j=1}^{\infty} jp_j.$$

The infinite sums can be truncated at appropriately large values to obtain good asymptotic approximations of behavior.

For smaller N, the fluid limit might deviate non-trivially from actual performance. We therefore note that, for a fixed N, one can compute the exact expected time to convergence directly using (backward) dynamic programming. If E_i represents the expected number of rounds until Y' reaches αN from i, and $q_{j,k}^N$ is the probability of Y' transitioning from state j to state k for the specific value of N, we have

$$E_i = 1 + \sum_{j=0}^{N-i} q_{i,i+j}E_{i+j}.$$

Note that $q_{j,k}$ can be determined exactly. Such calculations can be done in time $O(N^2)$, and very good approximations can be found even more quickly (near-linear time) by only considering jumps up to a certain amount (truncating the summation at $\min(c, N-i)$ for a constant c). Similar calculations can be performed to for example find the probability that convergence occurs within a specific number of rounds, keeping track of the probability of being in each state after each number of time steps. Let $P_{i,t}$ be the probability that it takes more than t rounds for Y' to reach αN from i. Then

$$P_{i,t} = \sum_{j=0}^{N-i} q_{i,i+j}P_{i+j,t-1}.$$

For a linear number of rounds, calculating the $P_{i,t}$ can be done in time $O(N^3)$.

Finally, we point out that standard tail bounds (e.g., Azuma's tail inequality, or tail bounds based on the mean field limit) can be used to show the number of time steps to reach convergence is tightly concentrated around its expectation with high probability. The only technical difficulty is that there is some small probability of a large jump in the number of transmitters choosing the right direction at each step; however, since the probability of k transmitters toggling their values when γ is constant falls even faster than geometrically in k, standard methods show that tight concentration still occurs.

VI. Improved Algorithms

We discuss a further approach to generalize our previous algorithm. Because of limited space, we consider this approach only briefly; further results will appear in future work.

Suppose that a node toggles its transmission, and learns via the feedback channel that the magnitude at the receiver increased during that round. The node then decides to lock that value with some probability w. Thus, in subsequent rounds, with probability $1-w$ the node will operate as before, but with probability w it will not flip a coin and simply resend the value that has been locked. The intuition for this approach is that if toggling the value at the ith transmitter improved the signal, then it is likely that X_i has the same sign as Y. Locking the value thus prevents X_i from being flipped to the incorrect value in later rounds. However, due to more than one node toggling in a given round, a node may actually lock the wrong value, making it harder to converge. The right tradeoff is not immediately clear, and hence depends on the target β.

This approach improves if unlocked transmitters correspondingly increase their probability of toggling each round. If the number of locked transmitters L were known, it would be natural to change the toggle probability from $p = \frac{\gamma}{N}$ to $p = \frac{\gamma}{N-L}$, keeping an average of γ toggles per round. But since the transmitters are distributed, they cannot determine L exactly. Here, we consider the following approximation: the transmitters count the number of successful rounds S where $Z[j] = +1$, and toggle with probability $p = \frac{\gamma}{N-Sw}$.

While this generalization can be studied in simulation, we also point out that both the dynamic programming and fluid limit approaches of Section V can be generalized to this more general class of algorithms, albeit with greater complexity as the number of locked transmitters and successful rounds must also be tracked.

VII. Simulations and Comparison

Simulations of the basic beamforming algorithm support the theoretical results in previous sections. Figure 3 shows a plot of the average number of rounds needed to achieve a received signal magnitude of βN as a function of N, for $\gamma = 1.0$ and three values of β. The curves confirm that the expected convergence time for the algorithm is linear in N.

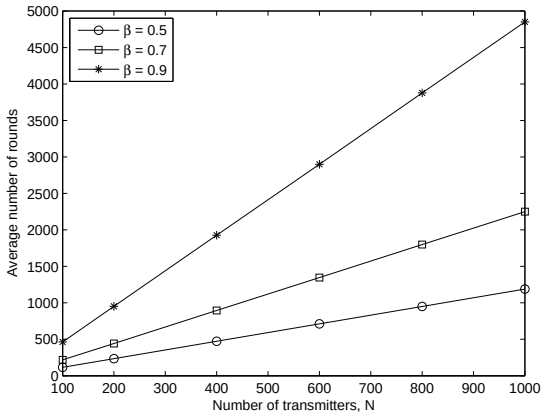

Fig. 3. The average number of rounds needed to achieve a signal magnitude of βN as a function of N, for three values of β.

In addition, it is possible to numerically optimize the value of γ. Figure 4 shows a plot of the average convergence time versus γ for $N = 1000$ and two values of β. From these curves, the optimum γ values (to the nearest tenth) are found

Upper bound	3273
Fluid limit approximation, $\gamma = 1.0$	2256
Dynamic programming, $\gamma = 1.0$	2247
Simulation of basic algorithm, $\gamma = 1.0$	2250
Simulation of "locking" algorithm, $w = 1.0$, $\gamma = 1.0$	1777
Fluid limit approximation, $\gamma = 2.2$	1869
Dynamic programming, $\gamma = 2.2$	1860
Simulation of basic algorithm, $\gamma = 2.2$	1865
Simulation of "locking" algorithm, $w = 0.7$, $\gamma = 2.2$	1265
Lower bound	119

TABLE I

NUMERICAL COMPARISON FOR $\beta = 0.7$ AND $N = 1000$

Upper bound	6259
Fluid limit approximation, $\gamma = 1.0$	4880
Dynamic programming, $\gamma = 1.0$	4850
Simulation of basic algorithm, $\gamma = 1.0$	4852
Simulation of "locking" algorithm, $w = 0.8$, $\gamma = 1.0$	3048
Fluid limit approximation, $\gamma = 1.5$	4594
Dynamic programming, $\gamma = 1.5$	4563
Simulation of basic algorithm, $\gamma = 1.5$	4565
Simulation of "locking" algorithm, $w = 0.8$, $\gamma = 1.5$	2223
Lower bound	531

TABLE II

NUMERICAL COMPARISON FOR $\beta = 0.9$ AND $N = 1000$

to be $\gamma^{opt} = 2.2$ for $\beta = 0.7$ and $\gamma^{opt} = 1.5$ for $\beta = 0.9$. Simulation results support the earlier intuition that the optimal value of γ is greater than 1, because $\gamma > 1$ increases the probability of a large jump in the Markov chain. Additional simulations showed that these optimum values are largely invariant to changes in N.

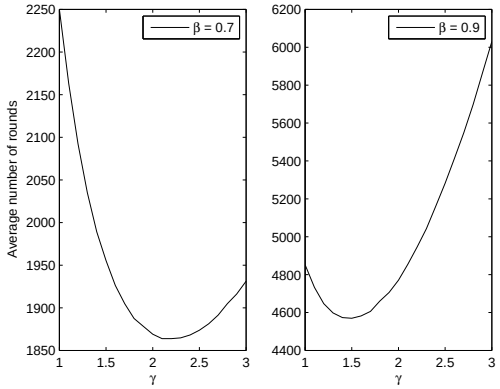

Fig. 4. The average number of rounds needed to achieve a signal magnitude of βN as a function of γ, for $\beta = 0.7$ (left) and $\beta = 0.9$ (right).

Table I compares the simulated convergence time of the basic algorithm from Section III, for $\beta = 0.7$ and $N = 1000$, to the upper and lower bounds, as well as the fluid limit approximation and dynamic programming. Results are provided both for $\gamma = 1.0$ and $\gamma^{opt} = 2.2$, and all results are rounded to the nearest integer. It also provides simulation results for the improved algorithm from Section VI, with an empirically optimized locking probability. Table II provides similar results for $\beta = 0.9$, where $\gamma^{opt} = 1.5$. While the two bounds are rather weak, the results are indeed close to the dynamic programming calculation, and the gains provided by optimizing γ and locking nodes are readily apparent. The fluid limit approximation is also quite accurate.

VIII. CONCLUSION

One main area of our future work is to generalize these approaches and results from the case of binary phases to the case of general phases. The combinatorial techniques can be extended to the problem with M discrete phases, which closely approximates continuous phases for large M. Moreover, even in the case of binary phases, there remain interesting open problems. While the difference between the upper and lower bounds on the number of rounds required for convergence is only a constant factor, such factors are quite significant in practice. Improving the upper bound via better algorithms and improving the lower bound via better analysis are both interesting in their own right and may shed light on more general cases. In particular, allowing time-varying parameters is an intriguing future direction.

In addition, while our fluid limit and dynamic programming approaches are computationally useful, they do not naturally give rise to formulae that can be used to predict behavior or allow us to optimize parameters algebraically. Finding simple closed forms (or at least good approximations) remains an important problem. Finally, we will also investigate the effect of noise on the beamforming problem, both in the receiver's measurement of signal magnitude and in the feedback channel.

ACKNOWLEDGMENT

The authors would like to thank Alex Dimakis for helpful discussions. Mark Johnson and Kannan Ramchandran were supported by NSF grants CCF-0635114 and CCF-072937. Michael Mitzenmacher was supported by NSF grants CCF-0634923 and CNS-0721491.

REFERENCES

[1] G. Barriac, R. Mudumbai, and U. Madhow, "Distributed beamforming for information transfer in sensor networks," in *International Conference on Information Processing in Sensor Networks (IPSN)*, Berkeley, CA, April 2004.

[2] H. Ochai, P. Mitran, H. Poor, and V. Tarokh, "Collaborative beamforming for distributed wireless ad hoc sensor networks," *IEEE Trans. on Signal Processing*, vol. 53, no. 11, Novemeber 2005.

[3] R. Mudumbai, J. Hespanha, U. Madhow, and G. Barriac, "Scalable feedback control for distributed beamforming in sensor networks," in *IEEE International Symposium on Information Theory (ISIT)*, Adelaide, Australia, September 2005.

[4] R. Mudumbai, G. Barriac, and U. Madhow, "On the feasibility of distributed beamforming in wireless networks," *IEEE Trans. on Wireless Communications*, vol. 6, no. 5, pp. 1754–1763, May 2007.

[5] J. Bucklew and W. Sethares, "Convergence of a class of decentralized beamforming algorithms," in *IEEE Workshop on Statistical Signal Processing (SSP)*, Madison, WI, August 2007.

[6] R. Mudumbai, B. Wild, U. Madhow, and K. Ramchandran, "Distributed beamforming using 1 bit feedback: From concept to realization," in *Allerton Conference on Communication, Control, and Computing*, Monticello, IL, September 2006.

[7] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, 1948.