Error Control Code Combining Techniques in Cluster-based Cooperative Wireless Networks

Su Yi*, Babak Azimi-Sadjadi*†, Shivkumar Kalyanaraman* and Vijaynarayanan Subramanian*
*Department of ECSE, Rensselaer Polytechnic Institute, TROY, NY 12180
† Institute for System Research, University of Maryland, College Park, MD 20742
yis@rpi.edu, {shivkuma, babak}@ecse.rpi.edu, vijay@networks.ecse.rpi.edu

***Abstract*— In this paper, we introduce a novel link layer cooperation technique in noisy wireless networks to improve overall system throughput and reliability, and to reduce the cost of retransmission and energy consumption. Under a cluster-based network design, code combining [2] is used together with FEC to improve the link layer reliability. This approach is different from how code combining is used in the conventional hybrid ARQ, which is in a sequential way. The analytical results and the simulations show that with the cooperation of nodes in a clustering network, the link reliability will be greatly improved with the same power consumption. Equivalently, this can be viewed as the same link performance with a lower transmission power and lower interference.**

I. Introduction

In this paper we present a new link layer cooperation scheme for multi-hop wireless networks and sensor networks to improve the overall channel quality for each transmitter/receiver pair. For this, we propose to extract diversity gain out of the redundancy inherently present in all broadcast network transmission, such as wireless networks, and direct those gains for chosen receiver nodes. The redundancy in such systems is present since the signal carried over such a channel is received (if not necessarily detected) by all nodes within transmission radius. Thus, in this distributed cooperative paradigm, packets are not relayed from one network node to the next, but from one cluster of nodes to the next cluster of nodes, until they reaches the destination.

Cooperation among nodes can be done in different communication layers. Fig.1 shows cooperation in the physical layer and in the link layer.

In the physical layer, cooperative nodes share their information to improve the channel quality using transmit and/or receive diversity (Fig.1a and 1b). Physical layer cooperation has been studied recently under the subject name of "cooperative diversity." In cooperative diversity the transmitting nodes use the nodes in the neighborhood of the transmitter and the receiver as relays [7], [8], [3], [9], active scatterers [6], or simply clusters of cooperating nodes [5], [1], to reduce the adverse effect of multipath fading in the wireless channel.

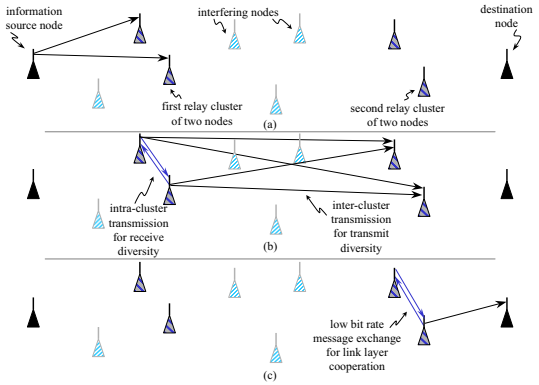

Fig. 1. Transmitting nodes group into cooperative clusters to relay the information from the source to the destination. (a) The information source reaches the first relay cluster. (b) The nodes in the relay cluster share their information for diversity gain. Then they relay the information to the next cluster. (c) The next cluster has a reliable channel with the destination node, hence there is no need of physical layer cooperation. A single node can relay the information to the final destination node.

In this paper we take a different approach and we use cooperation in the link layer. If the SNR of the received signal is moderately high, one can avoid physical layer cooperation to save on the bandwidth used for information sharing and synchronization [5], [1] and instead use the link layer cooperation to increase the overall throughput of the network. In the link layer cooperative transmission the cooperating nodes decode the received packets (instead of the individual bits/symbols done in the physical layer cooperation) and participate in the cooperative transmission of the error-free packets. The link layer cooperation can be implemented in two steps depending on the quality of the link:

Stage 1: Cluster head decides if cooperation is necessary. Unlike the node to node cooperative cluster transmission, a packet is successfully received if at least one node in the cluster receives the packet without error. The nodes with the error free packet send their status to the cluster head using a low bit rate message. The cluster head chooses one of the nodes with the error free packet to forward that packet to the next cluster.

Stage 2: Code combining and FEC. If no node receives the packet successfully, the cooperating nodes can combine their erroneous packets and use code combining techniques [2] to

This work is supported in part by NSF under contract number NSF-ITR 0313095, and by a grant from Intel Corp.

Babak Azimi-Sadjadi is supported in part by ARO under ODDR& EMURI97 Program Grant No. DAAG55-97-1-0114 to the Center for Dynamics and Control of Smart Structures (through Harvard University), and under ODDR&E MURI01 Program Grant No. DAAD19-01-1-0465 to the Center for Communicating Networked Control Systems (through Boston University), and by the NSF Learning and Intelligent Systems Initiative Grant CMS9720334.

reconstruct the packet. FEC can be designed over the entire frame to facilitate code combining. If the reconstruction is unsuccessful the master node sends an ARQ to the previous cluster for the packet retransmission.

The main technique in this paper is the use of the well-known code combining. In the conventional type I hybrid ARQ scheme with code combining, the repeated packets are sent upon each request. This retransmission based method can be considered a redundancy in time. In our new cooperative link layer paradigm, retransmission can be greatly reduced or avoided by making use of the wireless broadcast nature. In fact, the retransmission is replaced by information sharing among the nodes in the receiving cluster. In other words, we use the existing parallel channels between the transmitting node and the receiving nodes for code combining. This can be called redundancy in space. This method is well-suited for interactive real-time communication streams where waiting for retransmission introduces unacceptable delay and jitter. However, the cost for the node cooperation is the extra power and bandwidth used for the intra-cluster communication.

This paper is organized as follows: The performance analysis for the link layer cooperation is given in Section II. In Section III we present our simulations and results and in Section IV we give our concluding remarks and we lay out future work.

II. Performance Analysis for Link Layer Cooperation

A. Assumptions

We assume nodes are already clustered using some existing clustering protocol, and there are enough nodes in one cluster to cooperate. The packets in each cooperative node will be sent to the cluster head for code combining. So the number of repeated packets is identical to the number of cooperative nodes. Throughout the whole paper, L represents the number of nodes joining the cooperation. This is equivalent to the repeated packets in code combining.

In the cooperative cluster, the member nodes will transmit their received packets to the cluster head if necessary. The distance between the nodes in the cluster is much smaller than the distance between the transmitter and the receiver from different clusters. Therefore, the required intra-cluster transmission power is much smaller than the power of the inter-cluster transmission. In general the bit error rate for inter-cluster channel and intra-cluster channels are different. Let p_1 and p_0 be the bit error rate for the inter-cluster channel and intra-cluster channel, respectively. Therefore, a single bit traveling from the source to the cluster head via a member node, has the bit error probability equal to $p = p_1 + p_0 - p_1 p_0$.

B. Code Combining with Convolutional Codes in a Uniform Channel Condition

Code combing [2] represents a technique for combining L repeated packets encoded with a code of rate R to obtain a lower rate, R/L, and thus more powerful, error-correcting code, capable of allowing more channel errors. One feature of code combining is that the maximum-likelihood (ML) decoder will select the codeword m which maximizes the conditional probability between the received sequence $\mathbf{r}$ and the repeated codeword denoted by $\mathbf{v}_m$. Repeated codewords are transmitted over BSC channels with bit error rate p_i for $i = 1, 2, \cdots, L$. The decoding function can be written as

$$\max_m \left\{ p[\mathbf{r}|\mathbf{v}_m] = \prod_{i=1}^{L} (1-p_i)^{N-d_{mi}} p_i^{d_{mi}} \right\} \tag{1}$$

where d_{mi} is the number of bit disagreements for the ith codeword, and N is the pre-combined codeword length. An alternate way to write (1) is

$$\min_m \sum_{i=1}^{L} w_i d_{mi} \tag{2}$$

where weight (reliability factor) $w_i = \log \frac{1-p_i}{p_i}$.

If the cooperating nodes are close (relative to the distance between the transmitting node and the cluster head) to each other and close to the cluster head, the signal to noise ratios for all nodes are almost the same. In this case, the received packet weights w_i used in the code combining technique are the same for all the cooperative nodes, thus can be ignored. This scenario is referred as *uniform channel condition*.

If a block code is used for code combining, the complexity of the decoder depends greatly on the number of codewords. Therefore, to reduce the decoding complexity, we want the codeword length to be small. This will limit the use of block codes, since block codes are efficient in large blocks. For this reason, code combining is generally used for convolutional codes or for short block codes. For the rest of this section we analyze the performance of the code combining technique for convolutional codes. We adopt the notation used in [4].

For general convolutional codes with maximum likelihood decoding (Viterbi algorithm), the *bit error probability*, P_b, that is, the expected number of information bit errors per decoded information bit, is used to evaluate the performance of Viterbi algorithm. This bit error probability can be approximated by (upper bound):

$$P_b \approx B_{d_{free}} \left[2\sqrt{p(1-p)} \right]^{d_{free}} \tag{3}$$

where $B_{d_{free}}$ is the coefficient of $X^{d_{free}}$ in the bit *weight*[1] *enumerating function* (WEF) $B(X)$, and d_{free} is the *minimum free distance*.

In code combining the decoder receives L corrupted copies of the transmitted packets. A k-input n-output convolutional code with rate $R = k/n$ with L repeated packets, can be modelled by a k-input nL-output convolutional code with rate R/L. The Viterbi decoder for this rate R/L convolutional code has exactly the same trellis structure as the original rate R convolutional code. The only difference is how the metric for each branch of the trellis is calculated. Therefore,

[1]It is unfortunate that we use the term "weight" both for the measure of the quality of a link (w_i) and for the number of ones in a binary sequence (d or $W(\cdot)$).

the decoder for the code combiner and the decoder for the original convolutional code have the same order of complexity. Furthermore, it is easy to see that the WEF of the R/L rate convolutional code, $B_L(X)$, has the following relation with the WEF of the original code:

$$B_L(X) = B(X^L) \tag{4}$$

Hence the lowest power of X in $B_L(X)$ is Ld_{free}, i.e., $d_{free}(L) = Ld_{free}$, and

$$P_b(L) = B_{d_{free}} \left[2\sqrt{p(1-p)}\right]^{Ld_{free}} \tag{5}$$

In this expression, p refers to the transition probability of a BSC channel.

C. Code Combining with Different Channel Conditions

The assumption made in Section II-B is mainly valid when code combining is used together with hybrid ARQ, where the same channel is used for packet retransmission. However, in a cluster-based cooperation system, the channel condition can vary significantly among nodes. This is due to the different path losses caused by the different distances between receiver nodes and the transmitter. For this reason, the packets received with higher SNR should have higher weights in the decoder at the master node. The following part in this section will discuss the performance analysis of the weighted code combining. The results depend on the well-known performance bound for convolutional codes using Viterbi decoding, which is described in the following fact:

Fact 1: Using the analysis of the maximum-likelihood path selection on a trellis diagram, the error probability of a convolutional code with optimum decoding can be upper-bounded using a union bound, by the sum of the error probabilities of each of the paths. The bit-error probability, that is, the expected number of information bit errors per decoded information bit, can be approximated by:

$$P_b < \sum_{d=d_{free}}^{\infty} B_d P_d \tag{6}$$

B_d is the total number of nonzero information bits on all weight-d paths, divided by the number of information bits k per unit time (i.e., the coefficient of the weight-d term in the bit WEF $B(X) = \sum_{d=d_{free}}^{\infty} B_d X^d$ of the decoder). P_d is the event error probability for the weight-d path. This bound is tight, because P_d is very small. Therefore the union bound is the dominant part for the whole probability of error. $\diamond$

B_d is determined by the encoder. P_d has a nice expression for ordinary Viterbi decoding over a BSC channel. In weighted code combining, the result for P_d is more complicated. We assume that the decoder is aware of the channel condition for each cooperative node (this can be achieved by piggybacking extra bits during intra-cluster transmission process). Using the channel conditions, the decoder assigns the weight $w_i = \log \frac{1-p_i}{p_i}$ to the ith repeated packet according to the channel error rate p_i, for $i = 1, ..., L$.

A path with weight d would have the weight Ld when the code combining of order L is used. Let the *pseudo codeword* made of bits in these d positions for the correct path be $\mathbf{v}$, the corresponding pseudo codeword for the incorrect path be $\mathbf{v}'$, and the received set of packets be $\mathbf{r} = \{\mathbf{r}_1, \cdots, \mathbf{r}_L\}$. $\mathbf{r}_i$ is the ith received repeated packet. The path metric for $\mathbf{r}$ and $\mathbf{v}$ is given by

$$M(\mathbf{r}|\mathbf{v}) = \sum_{i=1}^{L} w_i d(\mathbf{r}_i, \mathbf{v}), \tag{7}$$

where $d(\mathbf{x}, \mathbf{y})$ is the Hamming distance between codewords $\mathbf{x}$ and $\mathbf{y}$.

For a weight-Ld path, a first event error will be made if, in the Ld positions in which the correct and incorrect path differ, the path metric for the incorrect path is less than that of the correct path (so the decoder wrongly chooses the incorrect path). The probability of such event is given by

$$P[M(\mathbf{r}|\mathbf{v}') < M(\mathbf{r}|\mathbf{v})] = P\left[\sum_{i=1}^{L} w_i d(\mathbf{r}_i, \mathbf{v}') < \sum_{i=1}^{L} w_i d(\mathbf{r}_i, \mathbf{v})\right].$$

From the linear property of the convolutional codes, the all-zero path is always assumed to be the correct path and the non all-zero path is the incorrect path. Therefore, $\mathbf{v}$ consists of d zeros and $\mathbf{v}'$ consists of d ones. Thus, $d(\mathbf{r}_i, \mathbf{v}) = W(\mathbf{r}_i)$ and $d(\mathbf{r}_i, \mathbf{v}') = d - W(\mathbf{r}_i)$, where $W(\mathbf{r})$ represents the Hamming weight of the received packet $\mathbf{r}$. So we have

$$\begin{aligned} P[M(\mathbf{r}|\mathbf{v}') < M(\mathbf{r}|\mathbf{v})] &= P\left[\sum_{i=1}^{L} w_i (d - 2W(\mathbf{r}_i)) < 0\right] \\ &= P\left[\sum_{i=1}^{L} w_i W(\mathbf{r}_i) > \frac{d}{2} \sum_{i=1}^{L} w_i\right]. \end{aligned}$$

If there is a tie between the metrics of the paths, decoder will randomly choose one. Let $c_{Ld} = \frac{d}{2} \sum_{i=1}^{L} w_i$, and $S = \sum_{i=1}^{L} w_i W(\mathbf{r}_i)$. Therefore, the probability of decoding error is given by

$$P_{Ld} = P[S > c_{Ld}] + \frac{1}{2} P[S = c_{Ld}] \tag{8}$$

S is the weighted sum of L binomial random variables with different parameter sets (d, p_i). We make use of the generating function to calculate the Probability Mass Function (PMF) of random variable S:

$$\begin{aligned} G_s(z) &= E\left[z^{\sum_{i=1}^{L} w_i W(\mathbf{r}_i)}\right] = \prod_{i=1}^{L} G_{W(\mathbf{r}_i)}(z^{w_i}) \\ &= \prod_{i=1}^{L} (1 - p_i + p_i z^{w_i})^d = \sum_k p_S(k) z^k \end{aligned} \tag{9}$$

The coefficient $p_S(k)$ is the probability of $S = k$. Therefore,

$$P_{Ld} = \sum_{k > c_{Ld}} p_S(k) + \frac{1}{2} p_S(c_{Ld}) \tag{10}$$

Thus, based on Fact 1 and (4), we have the following theorem:

Theorem 1: The upper bound for the bit-error probability of the distributed code combining method, P_b, is given by:

$$P_b < \sum_{d=d_{free}}^{\infty} B_d P_{Ld} \tag{11}$$

where B_d is the coefficient of the weight-d term in the bit WEF $B(X)$ of the original convolutional code, and B_{Ld} is given by (10). $\diamond$

Since p_i is small, P_{Ld} decreases greatly as d increases. P_b is generally dominated by the first several terms of the summation in (11), or even the first term $B_{d_{free}} P_{d_{free}}$. So this union bound is tight, and numerical results show the first several terms of the summation in (11) can be a good estimation of real P_b. There are $(d+1)^L$ terms in the right hand side of (9). For $L \leq 10$, the computation time of P_{Ld} is quite tolerable. Some results will be shown in the simulation section.

III. Simulations

In order to evaluate the performance of the cooperative networks, a set of random nodes representing the networks nodes are chosen according to the network topology as follows: the transmitter and the receiver cluster head are fixed nodes and are 250 meters apart. The cluster is formed around the cluster head in a circle with radius of 50 meters. The cooperative nodes are randomly placed as a uniform distribution inside the cluster. The topology of the simulated network is shown in Fig.2.

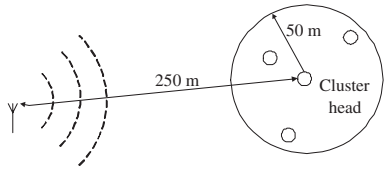

Fig. 2. Topology of the simulated network

In the following simulations the decoded bit-error rate P_b, is calculated using Theorem 1 from section II. Some simulation parameters are: path loss exponent - 3.5; channel model - Rayleigh fading; modulation - BPSK.

A. Link Layer Decoding Performance

We use different levels of power for inter-cluster and intra-cluster transmission because the distance between cluster nodes and the cluster head are at most 1/5 of the radio distance for inter-cluster transmission. Let PL represent the difference between the power used by the cluster nodes and the power at the sender node, in dB. We consider two cases where PL=10dB and 20dB, i.e. the cluster nodes use a transmit power that is 10dB and 20dB less than the sender transmit power, respectively. This means the SNR level is at least 4.5dB ($10\log(250/50)^{3.5} - 20$ dB = 4.5 dB) higher than the signal received from the sender. For each power level, the simulation takes 100 runs and finds the average decoded bit-error rate. A (2,1,3) convolutional code is used for code combining with Viterbi decoding at the cluster head. The decoded bit-error rate P_b with weighted code combining at the cluster head is plotted as a function of L in Fig.3. The SNR is measured at the receiver, i.e., the cluster head. Therefore the SNR is proportional to the sender transmission power. Changing PL from 10dB to 20dB does not change the overall performance of the code combining technique significantly. The change is negligible when the sender transmits at a considerably high power, e.g., in this simulation when SNR=8dB.

We also tried different cluster radii for the simulations. For PL=20dB, we simulated the cluster radii of 50m and 100m. The decoded bit-error rate is plotted in Fig.4. A larger cluster radius leads to a worse decoding performance since some cluster nodes may be too far from the sender node. However, it is shown in both Fig.3 and Fig.4 that the decoded bit error rate decreases sharply when L increases. A system designer should take this fact into account when deciding about the maximum number of the cooperation nodes.

B. Energy Consumption

To provide a reliable link performance, a very low bit error rate is desired. In another round of simulations, a couple of fixed decoded bit-error rates, 10^{-7}, 10^{-6}, and 10^{-5}, are set to be the objectives. The choice of the desired P_b mainly depends on the frame size. For each random topology, the sender power level is adjusted to achieve the desirable P_b. Cluster nodes use 20dB less power than the sender node (PL=20dB). We plot the required SNR at the cluster head as a function of cluster size to compare the dB gain of the cooperative code combining technique, as shown is Fig.5. Note when $L = 1$ it means there is no cooperation. So the difference between the SNR of cooperation and non-cooperation is very similar to the concept of *coding gain*.

The cost for the cooperation is the energy consumed at the cluster nodes. To take this into account, we also plot the aggregate energy spent in transmitter together with all the cluster nodes for successfully transmitting one bit. By successfully transmitting one bit we mean the residual bit error rate is less than 10^{-7}. The result of the normalized energy consumption (it takes one unit energy to successfully transmit one bit without cooperation) is shown in Fig.6. The simulation result of the energy consumption of a Hybrid ARQ scheme is included in this plot for comparison. In this case L represents the average number of repeated packets. For example, if the transmission power is adjusted as that it takes averagely 2 transmissions (1 retransmission) to successfully transmit one bit, then the total energy spent is .09 unit. This plot shows that the required transmitted energy decreases when the number of cluster nodes increases. Also cooperative code combining requires significantly less energy than the Hybrid ARQ scheme.

The above simulations are just some case studies to illustrate how cooperation can increase the decoding performance. If the channel quality is better than the channel used in these simulations, we may choose a code with a higher rate than 1/2 used in the above examples. In fact, such a low code rate as 1/2

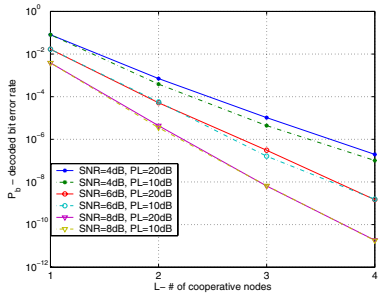

Fig. 3. Decoded bit-error rate P_b vs. number of cooperative nodes L. PL is the amount of power deduction of the intra-cluster transmission upon the inter-cluster transmission.

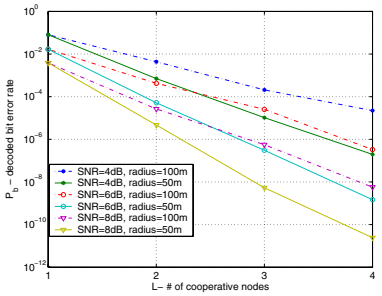

Fig. 4. Decoded bit-error rate P_b vs. number of cooperative nodes L with different cluster radius. Smaller cluster radius has a better performance.

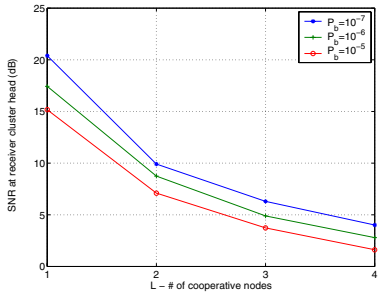

Fig. 5. SNR vs. number of cooperative nodes L. With a fixed objective P_b, the required SNR decreases with the increase of the cluster size L.

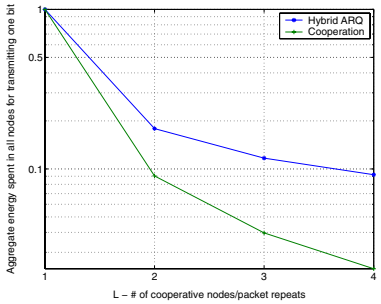

Fig. 6. Aggregate energy consumption vs. number of cooperative nodes or packet repeats L. A decoded bit-error rate $P_b = 10^{-7}$ is fixed.

will bring too much overhead in ad hoc networks. Obviously codes with lower rates have a better performance in terms of the decoded error rate. Given the desired P_b, and the channel condition, we can choose the appropriate operating point (code rate and cluster size) to meet the needs. A higher rate convolutional code can be achieved using punctured codes, which is a simple operation on a lower rate code without additional complexity. Likewise, the result can be extended to the case with longer distances between the transmitter and the receiver, different node density etc. The cluster size may be adapted to the channel condition and the code rate.

Due to the limited space, we intend to ignore some secondary considerations in this paper and leave them to our future publication. These considerations include the contribution of receptions and idle-listening on the RF channel to the energy consumption, and contention resolvent at the cluster head. We regard these issues as secondary because it is well accepted the energy consumed by reception is much less than that by transmission, and contention problem can be solved by intra-cluster transmission MAC, e.g., TDMA or CDMA.

IV. Conclusions and Future Work

In this paper we analyzed the decoding performance of the cluster-based cooperative networks with a code combining technique. Simulation results from various aspects show this cooperation architecture is effective in improving the link performance and reducing the energy consumption. This result is promising in that the reduced power requirement leads to less interference caused by a transmission, thus can improve the capacity of the wireless networks.

The results in this paper are under the consideration of a single hop network. Yet they are applicable to a multi-hop network as well. However, more problems will be involved, such as the effect of interference, MAC design, and so on. Our future work will look into the detailed cross layer design of the network, including cooperation-intended cluster-based routing, medium access issues in the intra-cluster communications, network performance from all aspects, and more information theoretic analysis of the coding technique and network capacity.

References

[1] B. Azimi-Sadjadi and A. Mercado. Diversity gain for cooperating nodes in multi-hop wireless networks. In *Proceedings of IEEE Vehicular Technology Conference 2004-Fall*, Los Angeles, CA, Sep 2004.

[2] D. Chase. Code combining–a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets. *IEEE Trans on Communications*, 33(5):385–393, May 1985.

[3] J. N. Laneman and D. N. C. Tse and G. W. Wornell. Cooperative Diversity in Wireless Networks: Efficient Protocols and Outage Behavior. *IEEE Trans. Inform. Theory*, 50(12), Dec 2003.

[4] Shu Lin and Daniel J. Costello. *Error Control Coding, 2nd Ed.* Pearson Education, 2004.

[5] A. Mercado and B. Azimi-Sadjadi. Power efficient link for multi-hop wireless networks. In *41st Annual Allerton Conference on Communication, Control, and Computing*, Oct 2003.

[6] A Scaglione and Y. Hong. Opportunistic large arrays: Cooperative transmission in wireless multihop ad hoc networks to reach far distances. *IEEE Transactions on Signal Processing*, 51(8):2082–2092, August 2003.

[7] A. Sendonaris, E. Erkip, and B. Aazhang. User Cooperation Diversity–Part I: System Description. *IEEE Transactions on Communications*, 51(11), November 2003.

[8] A. Sendonaris, E. Erkip, and B. Aazhang. User Cooperation Diversity–Part II: Implementation Aspects and Performance Analysis. *IEEE Transactions on Communications*, 51(11), November 2003.

[9] T. Hunter and A. Nosratinia. Cooperation Diversity Through Coding. *Proc. International Symposium on Information Theory*, June 2002.