Cluster Transmission Time Synchronization for Cooperative Transmission using Software Defined Radio

Yong Jun Chang
Georgia Institute of Technology
Email: yongjun.chang@gatech.edu

Mary Ann Ingram
Georgia Institute of Technology
Email: mai@gatech.edu

R. Scott Frazier
US Department of Defense
Email: rscottf@ieee.org

***Abstract*—We consider the time synchronization of concurrent transmissions from a cluster of cooperating software defined radios (SDRs). In this type of cooperative transmission (CT), each cooperating node derives its timing autonomously from a signal received from a source or another cluster. Timing errors, caused by SDR processing delays, cause excessively high delay spreads of the signal received at the destination node. To reduce these timing errors, we propose that relays use embedded time stamps to hold the packet for a fixed period before firing. The method has been implemented in an indoor non-line-of-sight channel, for a two-hop network (source-cluster-destination), using 64 kbps BFSK, non-coherent demodulation, and up to four diversity channels based on orthogonal carriers. The bit error rate (BER) and the measured cumulative distribution function (CDF) for the rms transmit time spreads (RTTSs) are presented, as functions of transmit power and number of cooperators. Mean RTTSs on the order of 50 ns are achieved. Also presented are the results of a "ping pong" experiment, in which a packet is transmitted back and forth between two clusters. CDFs of the RTTS for the first 10 hops are given, showing stable statistics after the second hop.**

I. Introduction

Cooperative transmission (CT) is a technique wherein one or more radios assist another radio, in the physical (PHY) layer, to transmit a single message, thereby forming a virtual multiple-input-single-output (VMISO) link. The signal-to-noise ratio (SNR) advantage from CT, which comes from array and spatial diversity gains of the transmit array, can be used to extend the range of the transmissions, reduce the radiated power from individual transmitters, or reduce the bit error rate (BER) of the received signal [1]. This paper reports experimental synchronization results for a type of CT in which the cooperating radios transmit at approximately the same time.

Different CT types have different synchronization requirements. Coherent CT requires channel state information (CSI) to be fed back from the receiver to each transmitter, so that the transmitters can be phase synchronized [2]. While this CT type has the highest gain, the network overhead to provide this feedback may not be possible in many applications. Non-coherent CT, the subject of this paper, requires no transmitter-side CSI, and provides gains of 10 dB and higher [1]. However, the signals must be transmitted in at least two orthogonal channels, and the receivers must be able to perform diversity combining of the soft outputs of those channels. Under a half duplex constraint (that is, a radio cannot transmit and receive at the same time), the orthogonal channels can be achieved in different ways. One way assumed by many authors is that each cooperating radio's transmission occurs in a different time slot [1][3][4]. This technique has the benefit that the receiver time synchronization is done separately for each cooperating transmitter, using the same techniques that are used for non-CT packet communications. The disadvantage of this approach is that the rate of the space-time code can be no greater than 1/2, because there are no concurrent transmissions allowed in a link. Alternatively, if the cooperative transmissions can take place synchronously, i.e. well within rms propagation delay spread of the channel, then a higher rate code, such as the Alamouti space-time block code (STBC) [5] might be used [6]. The opportunistic large array (OLA) is a synchronous type of CT where the number of cooperators is decided on the fly [7]; in theory, an OLA can use STBC or any other means for creating orthogonal channels. OLA has been considered as the basis for fast, mobility-tolerant routing in multi-hop networks, specifically for energy efficient broadcasts [7] and for reliable unicasts along cooperative routes[8].

In this paper, we show experimental results for synchronous CT, using binary frequency shift keying (BFSK), with non-coherent demodulation. Orthogonality is achieved in the frequency domain, by having different cooperating radios transmit on different orthogonal carriers. Because synchronous CT is not supported by any existing standard, we use software defined radios (SDRs) to implement our approach. We consider the two-hop scenario (source to cluster to destination) and the multi-cluster-hop scenario (source to cluster to cluster,. . ., to cluster, to destination). The main contributions of this paper are (1) to demonstrate CT in a practical office environment, and (2) to show measured two-hop rms transmit time spreads of on the order of 50 ns, and (3) to demonstrate experimentally that the time synchronization for the multi-cluster-hop scenario can remain stable as a function of hop number. The first two contributions show that synchronous CT can be practically achieved and that it has the potential to be used with broadband modulations like those in the IEEE standard 802.11a, which require overall delay spreads (CT plus channel) to be less than 800 ns. The last contribution shows that OLA-based unicasts

are possible, and that OLA-based broadcasts have promise. While the spectral efficiency (units: bits per second per Hz) of our type of CT is the same as the time-slotted type, the synchronous CT results reported here may be considered as an intermediate step toward achieving higher space-time coding rates.

II. Cluster Transmit Time Synchronization

Ideally, synchronous cooperating transmitters start their transmissions, or "fire," at the same time. However, several random phenomena cause the firing times to be slightly different, giving them a non-zero variance. In this section, we define variable times for these phenomena and describe our method for reducing the variance of the firing times.

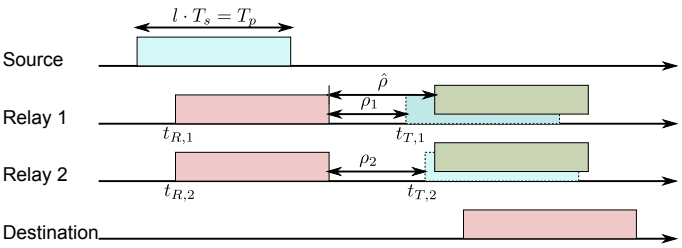


Fig. 1. Packet timing

A. Timing Model

Figure 1 shows a timing diagram for the two-hop scenario. The top trace represents the source packet, which is assumed to have the deterministic duration $T_p = l \cdot T_s$, where T_s is the source clock sample period and l is the number of samples in the source packet. Let $t_{R,i}$ be the the time that the source-transmitted waveform arrives at the antenna of the i_{th} relay node; this we call the "start of reception" (SOR). Our estimate of the value of $t_{R,i}$ is affected by the propagation time between the source and the i_{th} relay node, and the error in the estimate of $t_{R,i}$.

The next quantity affecting timing is ρ_i, which is the packet processing time for node i. ρ_i starts when the end of the source-transmitted packet arrives at the antenna and ends when the transmit (TX) first-in-first-out (FIFO) buffer is full. In the Universal Software Radio Peripheral (USRP1) and GNU-Radio, ρ_i contains the USB polling time and the processing time. The latter is not only large but also highly unpredictable because of buffering and scheduling in the Operation System (OS). Table I shows the minimum and maximum values of ρ in the GNURadio and the USRP system, measured for 1000 packets, as a function of buffer size and sample rate. The smallest variation is observed to be 3.3 ms, for Block Buffer of 4 KB and sample rate of 2.56 MHz. Therefore, if each relay is allowed to transmit when its TX FIFO is full, the random processing times will cause firing time variation of at least 3 ms, which is much too large to support broadband waveforms, such as OFDM with an 0.8 μs guard interval.

To avoid unpredictable latency caused by random processing times, we propose that all cooperating nodes wait to transmit

TABLE I
Measurement of ρ in GNURadio and USRP without using our method

Block Buffer	32KB		4KB	
Sample Rate	1.28M	2.56M	1.28M	2.56M
Min ρ	19.6ms	16.4ms	13.1ms	10.7ms
Max ρ	51.2ms	33.2ms	17.1ms	14.0ms

for a fixed period T_{proc} after the end of the source-transmitted packet arrives at the antenna. In symbols, the ith node will transmit at time $t_{R,i} + T_p + T_{proc}$. T_{proc} is selected so that $\alpha \times 100\%$ of the TX FIFOs will be full when the time comes to fire, where α is a design parameter such that $0 \leq \alpha \leq 1$. Therefore, T_{proc} may be expressed as

$$T_{proc}(\alpha) = \underset{t \in mT_s}{\arg\min} \{ Pr(\rho < t) \geq \alpha \}.$$

where $m \in \mathbb{Z}$. $T_{proc}(\alpha)$ should be determined prior to network operation.

Another concern is that because the different relays have different clocks, nodes with fast clocks will have T_{proc} expiring before the nodes with slow clocks. Each relay can estimate its own sampling clock offset by comparing the number of samples of a symbol duration with a designated oversampling rate. For example, if the designated oversampling rate is 10, and a node estimates that it has 11 samples per symbol, then it knows its clock is fast and it waits to fire until $T_{proc} \times (11/10)$ has expired.

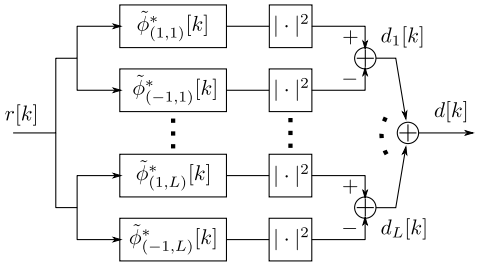


Fig. 2. Non-coherent BFSK demodulator with orthogonal frequency diversity

B. Estimating the Start of Reception (SOR)

Under ideal conditions, which implies non-CT, perfect carrier synchronization, and a static, flat-fading (or no-fading) channel, SOR estimation can be realized by cross-correlating the received signal with a known preamble, and locating the peak of the result. However, a carrier frequency offset (CFO) causes multiple peaks in the correlator output. Also a multipath channel or a previous-hop cluster transmission with multiple timing offsets will cause multiple peaks. When the peaks are of comparable height, noise can cause some relays to choose one peak and other relays to choose another peak, thereby inducing differences in the firing times of the relays in a cluster. To avoid these problems, we propose finding the mean location of the windowed correlation output. The baseband signal of BFSK in the l_{th} orthogonal channel with frequency separation

$\triangle f$ is as follows:

$$s_m^l(t) = \sqrt{\frac{2\mathcal{E}}{T}} e^{j\pi\{(m(1/2)+l)\triangle f\}t} \quad (1)$$

where m denotes the symbol $\in \{-1, 1\}$ and $\mathcal{E}$ is transmit symbol energy. The received signal superimposing L orthogonal transmissions through complex flat fading channel gains h_l and propagation delays τ_l, $1 \leq l \leq L$, is given by

$$r(t) = \sum_{l=1}^{L} h_l s_m^l(t - \tau_l) + w(t) \quad (2)$$

where $w(t)$ is complex white Gaussian noise. At the receiver, the digitized $r(t)$ passes through L pairs of BFSK envelope detectors which consist of matched filter banks and squarers as shown in Figure 2. The outputs, $d_i[k]$, of the L diversity channels are combined to produce the bipolar, soft-valued sequence $d[k]$. The first K symbols of the packet compose the BFSK preamble $p[k]$, $\{0 \leq k < K\}$. The correlator output is

$$\Omega[n] = \sum_l d[l] p\big[\lfloor (l-n)/S \rfloor\big] \quad (3)$$

where S is the number of samples per symbol. We can define the windowed correlator output as

$$\Omega_w[n] = \begin{cases} \Omega[n] & \text{if signum}(d(n+kS)) = p[k] \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $0 \leq k < K$. Now the SOR estimate is

$$\hat{\xi} = \frac{\sum n\Omega_w[n]}{\sum \Omega_w[n]}. \quad (5)$$

III. Implementation on GNURadio and USRP

In our implementation, a wireless node consists of an RF daughterboard, an USRP1, and GNU Radio running on a personal computer (PC). The USRP1 has an ADC/DAC and an FPGA to digitize the analog waveform and create the baseband sampled data stream. The PC does all the baseband signal processing; for instance, it implements the blocks in Fig. 2 and detects the data symbols. The FPGA has a sampling clock at a frequency of $f_s = 1/T_s$ = 64 MHz, but there is no counter in the default configuration. Because the transmit FIFO is on the FPGA, the relay must fire according to this clock. However, the firing time is determined relative to the start of packet reception (SOR), which is detected in the PC, not the FPGA. Because the PC and the FPGA are connected by a USB cable, the PC's clock is not related to the FPGA's clock. In other words, if the PC were to decide that the time to fire is "now!" the FPGA would get the command some long and random time later, because of the slow and uncertain timing of the USB transmissions.

Our solution to this problem is to implement on the FPGA a 32-bit hardware counter that increments every clock cycle, to measure the time, and for the FPGA to insert a time tag after every Gth data sample. Since the time tag looks like data from the point of view of the PC, we let the least significant bit (LSB) of the data be an indicator of the time tags. Specifically, the LSB is always zero, except when the data is a time tag- then the LSB is set to one. As shown in Figure 3, when the data comes into the PC from the USB, the splitter looks at the LSBs and separates the time tags from the data. However, each time tag still remains associated with its particular data sample. This way, when the PC finds the SOR, it can interpolate between the nearest time tags to learn what FPGA time should be associated with the SOR. To this SOR time, the PC adds the packet time, T_p, and the holding time, T_{proc}, to get the "Transmit Time-tag", Tg. Before the packet is sent to the USRP1 via the USB, Tg is inserted at the beginning of the data stream and is indicated by the LSB. The "Gate Block" in the FPGA finds Tg by reading the LSBs, and holds the packet if the FPGA clock counter is less than Tg. The transmission will be released when the FPGA clock counter is the same as Tg. If Tg has expired, the gate block will flush out the entire packet or partially abandon a few samples. The rollover of the time-tag will not be considered, because the total period of the 32-bit counter is much longer than the duration of the packet.

IV. Experimental Setup

A. GNURadio Architecture

Our PCs have 2.2Ghz Celeron Processors and 2GB of RAM. All baseband signal blocks, including the proposed methods, are written in C++ and Python languages, as shown in Figure 3. Our BFSK data rate is 64 kbps and the sample rate is 1 MHz, implying that the number of samples per symbol is $S = 16$. We choose $\alpha = 1$, which means that T_{proc} should be long enough so that all relays are ready to fire by the time Tg matches the clock counter. Based on Table 1, for a block buffer of 4KB, $T_{proc} = 20ms$ is sufficient.

B. RMS Transmit Time Spread Measurement

The performance of time synchronization is evaluated by measuring transmission time differences between relay nodes within the same relaying cluster. When Tg matches to the 32-bit hardware counter, the relay fires and simultaneously the MUX control signal on the USRP1 switches states. We connect this MUX control signal logically to the external general purpose I/O (GPIO), which in turn, is connected physically by wire to a customized FPGA board that we call an observer device. We stress that the observer device, which receives trigger signals by wire from up to four nodes, does not control when the radios fire. It only records the trigger times so that we can precisely determine the quality of CT transmit time synchronization. The timing resolution of the observer device is 15.625 ns. We define the rms transmit time spread (RTTS) of the l_{th} packet as

$$\sigma_{\tau,l} = \sqrt{\frac{\sum_i^N (t_{T,i}^{(l)} - \hat{t}_T^{(l)})^2}{N-1}}$$

where $t_{T,i}^{(l)}$ is a measured transmission time of l_{th} packet of i_{th} relay node and $\hat{t}_T^{(l)}$ is the sample mean of transmission time of l_{th} packet. The RTTS data is averaged over trials and channel realizations.

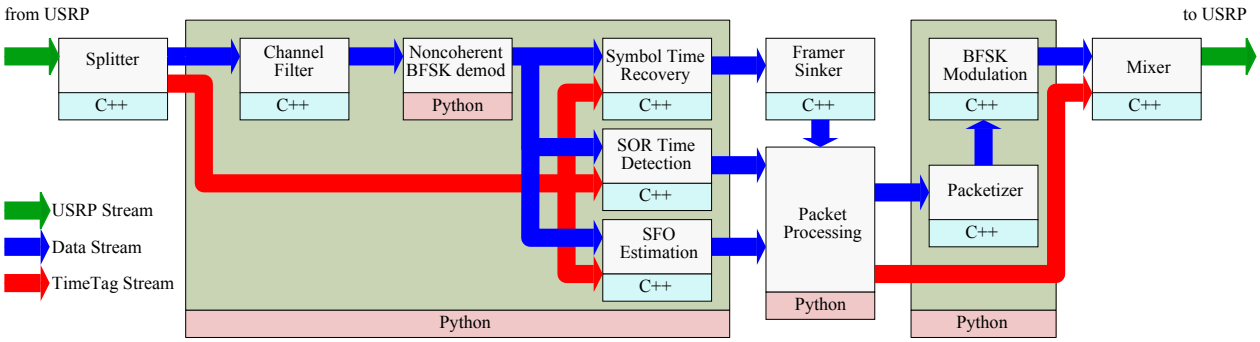


Fig. 3. Block diagram of single node for supporting transmission time synchronization

C. BER Measurement

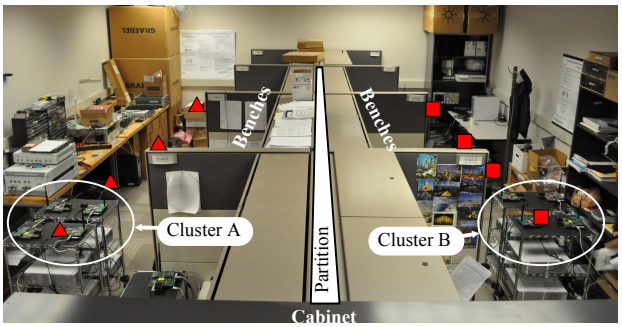
Diversity performance is indicated by the slope of the curve of BER vs. SNR [9]. In order to get an experimental BER curve, we have to know the average SNR for a given topology of a network. Many factors are involved with an accurate SNR estimation (i.e. transmit power, distance, channel, residual error of CFO and SFO.) To simplify our experiment, we elect to vary transmit power and fix the distance between nodes.

Variation in the instantaneous SNR from multipath fading is overcome by averaging BER data over various channel realizations. In this experiment, two different carrier frequencies 2.482Ghz and 2.492Ghz, and different transmitter and receiver locations, keeping the distance constant, are used to realize channel variations.

In practical radio systems, it is easier to measure packet error rate (PER) rather than BER. The CRC error check determines if the received packet is corrupted. The PER can be obtained if there is prior knowledge of how many packets are sent by a source. The BER can be estimated by

$$BER = 1 - (1 - PER)^{\frac{1}{D}},$$

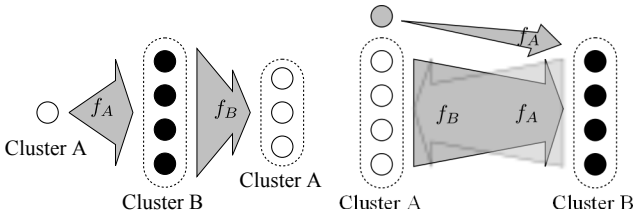
where D is the number of bits in a packet.



Bench	30"H	Engineered Wood
Cabinet	72"H	Metal
Partition	65"H	Foam-padded with metal frame

▲ Location of Cluster A
■ Location of Cluster B
* Avg. Distance = 5m

Fig. 4. GNURadio node placement



(a) Two-hop experiment (b) Ping pong experiment

Fig. 5. The two-hop and ping pong experiments

D. Node Placement and Testing Sequence

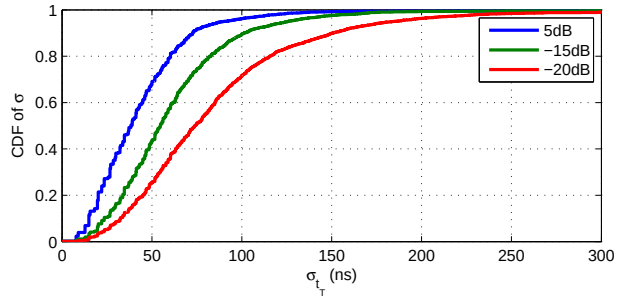
The measurements were taken in the Smart Antenna Research Laboratory of Centergy Building, Georgia Institute of Technology. The layout of the lab and the wireless node placements are shown in Figure 4. Two clusters were used to perform each experiment, and each cluster is composed of four GNURadio nodes and an observer device. The experiment data was collected at four different locations, as indicated by the red triangles and squares in the figure. The scenario of the two-hop CT experiment is depicted in Figure 5(a). A node in Cluster A is set as the source node. The source node transmits a message to Cluster B through carrier frequency f_A. While all the nodes in Cluster B should be configured to listen to frequency f_A, rest of nodes in cluster A should be able to listen only to frequency f_B. Different frequencies are used only to provide isolation, since the source and destination nodes are on the same cart. The nodes in Cluster B relay the source message through frequency f_B. At time of relaying, transmit time differences between relay nodes in Cluster B were measured and recorded by using the observer device.

By changing the number of relaying nodes in Cluster B and the transmit power of each node, RTTS of Cluster B and BER performance of rest of the nodes in Cluster A were measured.

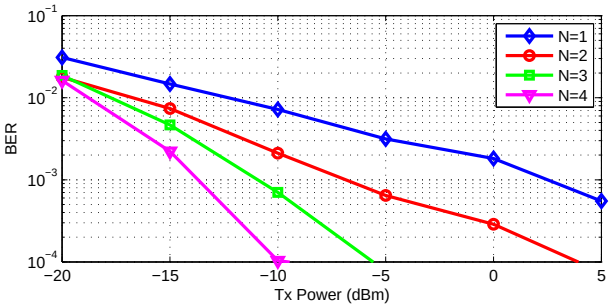
The results of a second, “ping pong”, experiment are shown in Figure 5(b). An initial source node is located near the Cluster A and transmits a single packet through frequency f_B. This single packet is transmitted back and forth between the clusters. The RTTS of each successive hop is measured and recorded.

The GNURadio and USRP system do not provide gain control of the transmit amplifier. Only the amplitude of the

digitized samples can be controlled. We use the range of 500-10000 out of 32768, because below 500, there is too much quantization error. We calibrated the transmit power out of the antenna connector using a spectrum analyzer. The SDR provides a code-controlled low-noise amplifier (LNA) gain in the range of 0-90dB. We set this gain to 35dB which avoids receiver saturation at 5m distance with 10dB margin. Therefore the "Tx power" in the figures can be considered as normalized by the 55dB of receiver gain.



(a) CDF of rms transmit time spread where $N = 4$



(b) BER versus transmit power at the three destination nodes

Fig. 6. Experimental rms transmit time spread and BER curve

V. Experimental Results

Figure 6(a) provides the experimental RTTS versus transmit power at the nodes in cluster B (i.e. relay nodes). In Figure 6(b), we observe that the RTTS decreases as SNR increases. It should be noted that the RTTS is plotted against transmit power rather than SNR.

Figure 7 shows the result of 500 trials of the "ping pong" experiment with 10 hops per trial. The result shows that the RTTS of clusters do not increase as the number of hops increases. In Figure 7, the better RTTS performance of the second hop, compared to the further hops, is not surprising, as the reference signal of SOR detection is a single source. The graph shows that RTTS at 3-10th hops have approximately same CDFs.

VI. Conclusion

In this paper, the cluster transmit time synchronization of cooperating nodes was proposed and measured results of OLA-based CT were described. It was shown that the cooperative diversity can be achieved without wired synchronization and that CT improves the overall BER performance.

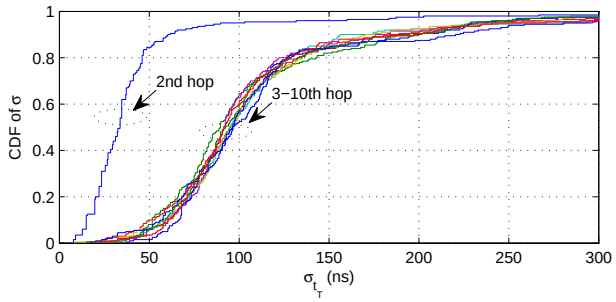


Fig. 7. Experimental RMS transmit time spread (RTTS)

The implementation using SDRs shows that OLA-based CT, whose benefits have been shown in theory and simulations, is practical. Our ongoing work includes exploration of more bandwidth efficient diversity schemes, such as distributed STBC, and OLA-based network layer protocols [8] using this testbed.

References

[1] J. Laneman, D. Tse, and G. Wornell, "Cooperative diversity in wireless networks: Efficient protocols and outage behavior," *IEEE Transactions on Information Theory*, vol. 50, no. 12, pp. 3062–3080, Dec. 2004.

[2] R. Mudumbai, D. Brown, U. Madhow, and H. Poor, "Distributed transmit beamforming: challenges and recent progress," *IEEE Communications Magazine*, vol. 47, no. 2, pp. 102–110, February 2009.

[3] A. Nosratinia, T. Hunter, and A. Hedayat, "Cooperative communication in wireless networks," *IEEE Communications Magazine*, vol. 42, no. 10, pp. 74–80, Oct. 2004.

[4] G. Bradford and J. Laneman, "An experimental framework for the evaluation of cooperative diversity," in *CISS 43rd Annual Conference on Information Sciences and Systems*, March 2009, pp. 641–645.

[5] S. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, Oct 1998.

[6] J. Laneman and G. Wornell, "Distributed space-time-coded protocols for exploiting cooperative diversity in wireless networks," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2415–2425, Oct. 2003.

[7] A. Scaglione and Y.-W. Hong, "Opportunistic large arrays: cooperative transmission in wireless multihop ad hoc networks to reach far distances," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2082–2092, Aug. 2003.

[8] L. Thanayankizil, A. Kailas, and M. A. Ingram, "Routing for wireless sensor networks with an opportunistic large array (ola) physical layer," *Ad Hoc & Sensor Wireless Networks, Special Issue on Sensor Technologies and Applications*, vol. 8, no. 1-2, pp. 79–117, 2009.

[9] L. Zheng and D. Tse, "Diversity and multiplexing: a fundamental tradeoff in multiple-antenna channels," *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.