Computation in Noisy Radio Networks

Eyal Kushilevitz* Yishay Mansour†

Abstract

In this paper we examine noisy radio (broadcast) networks in which every bit transmitted has a certain probability to be flipped. Each processor has some initial input bit, and the goal is to compute a function of the initial inputs. In this model we show a protocol to compute any threshold function using only a linear number of transmissions.

1 Introduction

The influence of *noise* (or faults) on the complexity of computation was studied in many contexts. In particular people were interested in *random* noise. In a typical such scenario, it is assumed that the outcome of each operation is noisy with some fixed probability p and all the faults are independent. Usually, if t is the number of operations performed by the computation, then by repeating each operation $O(\log t)$ times and taking the majority of the results, one can ensure a constant probability of error at the cost of $O(t \log t)$ operations. It is desirable however to obtain a cost of $O(t)$ (i.e., increase only by a constant factor the resources utilized compared to the non-noisy case). In spite of the apparent simplicity of this noise model, it turns out that overcoming such a noise with only a constant increase in the cost may be non-trivial and in some cases impossible. Such a noise model was studied in the context of circuits with noisy gates (e.g., [N56, DO77a, DO77b, Pip85, Gál91, RS91]), decision trees with noisy nodes (e.g., [FPRU90, RS91, EP96]), communication complexity ([Sch92, Sch93, RS94]), and others (e.g., [Tay68, Kuz73, Gác86, Spi96]).

In this paper we consider noisy radio (broadcast) networks. The main feature of radio networks is that when a processor sends a message all its neighbors receive it. There has been a considerable amount of attention given to this model. The most important problem studied is the broadcast problem, where one processor wants to propagate some message to all the other processors (e.g., [CK85, CK87, CW87, ABLP91, BGI92, KM93, GM95] and references therein). In addition, general transformations were developed to translate protocols from the radio model to the standard point to point network model [ABLP92]. All the above work assumes noise-free transmission. We consider random noise as described above. To simplify things, we ignore the topology of the network and study a fully connected network; this allows us to focus on the issue of efficiently overcoming the noise.

This question of noisy radio (broadcast) networks was already studied by Gallager [Gal88]. More formally, Gallager considers the following setting: there are n processors in the network $P_1, \ldots, P_n$. Each processor P_i starts the protocol with an input bit b_i and their goal is to compute the value of some function $f(b_1, \ldots, b_n)$. The basic operation in this model is a *broadcast* operation in which one processor P_i sends a bit b and all processors hear it. We rule out encoding of information using silence by assuming that a processor has to send a bit, whenever it is its turn to broadcast. Conceptually, one may view silence and non-silence as encoding the values zero and one, and therefore they too should be subject to noise. (See a detailed discussion in Section 2.2.)

Obviously, in the noise-free case every function can be computed using n broadcast operations and n is also a lower bound for non-degenerate functions. In the *noisy* setting, when P_i broadcasts a bit every processor P_j receives this bit with probability $1-p$ and with probability p it receives its complement, $\bar{b}$. The question is what bounds can be proved in the noisy case. By the above arguments, $O(n \log n)$ broadcast operations suffice and $\Omega(n)$ are necessary. Gallager [Gal88] proved that in fact every function can be computed using $O(n \log \log n)$ broadcast operations.[1] It remained open whether one can achieve the desired $O(n)$ upper bound. In particular, Yao [Yao97] suggested to concentrate on this open problem with respect to the family of *threshold functions*, as a special case which already

*Dept. of Computer Science, Technion, Haifa 32000, Israel. e-mail: eyalk@cs.technion.ac.il . http://www.cs.technion.ac.il/~eyalk .

†Dept. of Computer Science, Tel-Aviv University. e-mail: mansour@math.tau.ac.il .

[1] Gallager's paper (and its title) [Gal88] concentrate on the *parity* function; however, the protocol presented there can in fact deal with any function.

seems to be difficult.[2]

In this paper we answer this question in the affirmative for all *threshold functions* (this includes as special cases the functions AND, OR and $MAJORITY$). Namely, we present a protocol to compute any such function using $O(n)$ broadcast operations. The correctness proof of our protocol uses some probabilistic analysis that might be useful in other settings as well. In particular, in our protocol the processors need not have an a-priory knowledge of p, the noise probability (although for simplifying the presentation we start with the assumption that p is known; later however we show how to get rid of this assumption and use any upper bound $p' < 1/2$ on the true noise rate p.). Also, as is usually required from protocols that run in noisy environments, the schedule in our protocol (i.e., the decision which processor broadcasts next) is *oblivious*; i.e., it does not depend on messages received during the execution of the protocol (which are potentially noisy).

Organization: In Section 2 we provide some necessary background from probability theory together with a precise definition of the model. In Section 3 we present our protocol. We start with the slightly easier case where the noise rate is known to all processors. Later, in Section 4 we extend our solution to handle the case where p is unknown.

2 Preliminaries

2.1 Probability Theory

In this section we provide some notations and facts from probability theory. Most of these are standard but certain facts are less commonly used in computer science.

Denote by $b(k;n,p)$ the probability of having *exactly* k successes in n trials where each trial has probability p for success. That is,

$$b(k;n,p) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Denote by $B(k;n,p)$ the probability of having *at most* k successes in n trials where each trial has probability p for success. That is,

$$B(k;n,p) = \sum_{i=0}^{k} b(i;n,p) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}.$$

For our proofs we will need to estimate such binomial coefficients. Below are some useful (standard) facts that will help us in doing so (see, e.g., [SF96], page 169). We start with Stirling formula

$$n! = \sqrt{2\pi n}(\frac{n}{e})^n (1 + O(\frac{1}{n})).$$

Let k be either $\lfloor pn \rfloor$ or $\lceil pn \rceil$ (i.e., k is approximately the expected number of successes), then,

$$(1) \qquad b(k;n,p) = \frac{1}{\sqrt{2\pi p(1-p)n}}(1 + O(\frac{1}{n}))$$

Finally note that $b(k;n,p)$ is maximal for these values of k. That is, for any $k \leq \lfloor pn \rfloor$,

$$(2) \qquad b(k-1;n,p) \leq b(k;n,p)$$

and similarly, for any $k \geq \lceil pn \rceil$,

$$b(k+1;n,p) \leq b(k;n,p).$$

The following inequality, due to Chernoff [Che52], is a standard and useful fact. Let $X_1, \ldots, X_n$ be n independent 0-1 random variables with "success" probability p (i.e., $Pr[X_i = 1] = p$). Then, the probability that $\sum_{i=1}^{n} X_i$ deviates significantly from its expected value (i.e., pn) is small. More precisely,

Lemma 2.1 *Let X_i be n i.i.d. binary random variables, and $p = Pr[X_i = 1]$. Then,*

$$Pr[\sum_{i=1}^{n} X_i < (p-\lambda)n] \quad < \quad e^{-2\lambda^2 n}.$$

Suppose that we have n independent 0-1 random variables $X_1, \ldots, X_n$ with "success" probabilities $p_1, \ldots, p_n$ (respectively). Obviously, the expected value of $\sum_{i=1}^{n} X_i$ is $\mu \stackrel{\triangle}{=} \sum_{i=1}^{n} p_i$. Consider all possible choices for $p_1, \ldots, p_n$ such that their sum is μ. The following lemma, due to Hoeffding [Hof56], states that the probability of getting k successes (for k which is smaller than the expectation) is maximal if all the p_i's are equal (i.e., each of them is μ/n). Formally:

Lemma 2.2 *Let X_i $(1 \leq i \leq n)$ be n independent, binary random variables. Let $\mu = E[\sum_{i=1}^{n} X_i]$. For an integer $k \leq \mu - 1$,*

$$Pr[\sum_{i=1}^{n} X_i \leq k] \quad \leq \quad B(k;n,\mu/n)$$

This lemma allows transforming situations where we have several types of random variables with different probabilities, into a more "uniform" setting.

The *median* of a probability distribution is a number x such that $Pr(X \leq x) \geq 1/2$ and $Pr[X \geq x] \geq 1/2$. The following lemma [JS68] states that the median of a binomial distribution is very close to its expectation.

[2] A threshold function is defined using a parameter k. It returns 1 if $\sum_{i=1}^{n} b_i \geq k$ and 0 otherwise.

Lemma 2.3 *Let X_i be n independent identically distributed binary random variables. Let $\mu = E[\sum_{i=1}^{n} X_i]$. The median of the distribution, is either $\lfloor \mu \rfloor$ or $\lceil \mu \rceil$.*

We also use the following property of $b(k; n, p)$ that states that when k is close to the expectation (i.e., pn) its value is about $1/\sqrt{n}$ (for completeness we prove this in the Appendix).

Lemma 2.4 *Let c and p be constants. Then, for sufficiently large n,*

$$b(\lfloor np \rfloor - c; n, p) = \Theta(\frac{1}{\sqrt{n}}).$$

2.2 Model

The model consists of n processors $P_1, \ldots, P_n$ that are communicating over a (single-hop) radio network. When a certain processor P_i broadcasts a bit, all the other processors (for simplicity, including itself) receive the bit corrupted by some noise. Formally, denote by $b_{i,k}$ the k-th bit submitted by P_i during the execution of the protocol. When processor P_i sends this bit $b_{i,k}$, processor P_j receives, $b_{i,k} \oplus r_{i,j,k}$ where the $r_{i,j,k}$'s are independent and identically distributed (i.i.d), binary random variables such that $Pr[r_{i,j,k} = 1] = p < 1/2$.

Initially, each processor P_i has an input bit b_i. The goal of $P_1, \ldots, P_n$ is to compute some value of a function f on their inputs (i.e., $f(b_1, \ldots, b_n)$). We will be interested mainly in threshold functions, i.e., functions that depend only on whether the number of ones in the input exceeds a given threshold. We say that the protocol is *correct* if with probability $\geq 1-\epsilon$ *all* the processors output the correct value (where ϵ is any fixed constant; e.g., $\epsilon = 0.01$).

We also require that the scheduling of the transmissions will be *oblivious*; i.e., it can be fixed ahead of time independently of the input (and of previous transmissions).[3] Whenever it is P_i's turn to broadcast according to the schedule it *must* send a bit and is not allowed to remain silent.[4]

[3] This is a common requirement for communication in noisy environments. The reason behind this requirement is that if the schedule depends on previous transmissions then, since due to the noise different processors get different information, we can get into chaotic situations.

[4] There are two widely-studied models of broadcast/radio networks (see, e.g., [Tan81] for a discussion including a technological justification for each of the models): (a) a model that allows a processor to remain silent in its turn; and (b) a model that does not allow a processor to remain silent in its turn. Our work, as well as [Gal88], and many others refer to the latter model. It is easy to see that in the former model all functions can be computed in $O(n)$ broadcasts even in the presence of noise (the reason is that in this model processors can encode information by silence and silence is assumed not to be effected by noise).

The communication complexity measure is the number of broadcast operations performed (as a function of n). In particular, the obliviousness of the schedule implies that the number of transmissions is the same for all inputs and in all possible executions.

3 Computing a Threshold

In this section we present a protocol, `THRESHOLD`(k, ϵ), which allows the n processors to decide whether the number of ones in the input, $\ell = \sum_{i=1}^{n} b_i$, is at least as large as the "threshold" parameter k or not. If $\ell \geq k$ then the output should be 1 (i.e., "TRUE"), else ($\ell < k$) the output should be 0 (i.e., "FALSE"). We denote by ϵ the accuracy parameter. Recall that the correctness condition requires that with probability at least $1-\epsilon$ *all* the processors have the correct value. We first assume that p, the noise rate, is known to all processors. Later we show how to remove this assumption (Section 4).

The basic idea is that each processor broadcasts its bit b_i, and then compares the number of ones that it receives to the expected number of ones, given that the input had exactly k ones (to make sure that for different values of ℓ the expected number of ones is significantly different, we repeat each broadcast several times). We denote by β_i the result of this comparison in processor P_i. Then, by appropriately taking majorities we compute the function. One of the difficulties is that when we compute the majorities the transmissions may, again, be corrupted by noise.

At a high level the process may be viewed as follows. We start with a small bias of the inputs to the correct value, a bias of $1/n$. After the first round of communication the bias grows to $1/\sqrt{n}$. After the second round the bias (for each particular processor) is already a constant. Finally after the third round, all processors should have the right value. Clearly, due to the noise, each step in this description might fail. In the analysis, we compute the failure probability of each round, and sum them up in order to get the total probability of failure. This total probability of failure should be less than our accuracy parameter, ϵ.

The protocol `THRESHOLD(k,`ϵ`)` for processor P_i is as follows. (Recall that each time that we send a bit to all processor, it is also sent to the sending processor.)

Initialize Given p, n and k,
Let $\theta_k = pnm_2 + 2m_1k - m_1$,
$m_1 = \max\{\ln(3/\epsilon)/(2c_2^2), (c_2(1-2p))^{-2}\}$
and $m_2 = 2m_1/(1-2p)$
(where c_2 is a constant to be fixed latter).
Round A Send input b_i to all processors m_2 times.
Receive $m_2 n$ bits $\alpha_{i,j}$.
Let $A_i = \sum_j \alpha_{i,j}$.
Let $\beta_i = 1$ iff $A_i \geq \theta_k$.
Round B Send to all processors β_i once.
Let γ_i be the majority of the bits received.
Round C Send to all processors γ_i once.
OUTPUT The majority of the bits received.

3.1 Analysis

The following theorem is obvious from the code of the procedure.

Theorem 3.1 THRESHOLD(k, ϵ) *is an oblivious protocol that requires only $O(n)$ broadcast operations.*

Our main task is to compute the error probability of the procedure. In the following series of lemmi we establish the desired property after each round, and latter in Theorem 3.7 we combine these properties to establish that the error is bounded by ϵ.

Lemma 3.2 *Let $\ell = \sum_i b_i$. Then,*
(1) The random variables $A_1, \ldots, A_n$ are i.i.d; and
(2) For all i, $E[A_i] = m_2 pn + 2m_1\ell$.

Proof: The random variables A_i are identical, since for every processor A_i is defined in the same way (in particular the noise is the same for all messages, and for convenience every processor sends its input bit to itself as well). The independence is by the independence of the noise in our model.

As for the expectation: each of the ℓ bits b_j such that $b_j = 1$ contributes to the expectation $1-p$ in each of the $m_2 = 2m_1/(1-2p)$ times it is transmitted (since $1-p$ is the probability that a "1" bit is received as "1"), and each of the $n-\ell$ bits b_j such that $b_j = 0$ contributes to the expectation p in each of the m_2 times it is transmitted (since p is the probability that a "0" bit is received as "1"). All together

$$\begin{aligned} E[A_i] &= m_2(\ell(1-p) + (n-\ell)p) \\ &= m_2(\ell(1-2p) + np) \\ &= npm_2 + 2m_1\ell, \end{aligned}$$

where the last equality follows from the definition of m_2. □

Note that we chose the value θ_k such that it is at least m_1 away from the expected value of A_i. The next lemma claims that by comparing A_i and θ_k we can know, with probability slightly better than 1/2, whether ℓ (the number of 1's in the input) is at least k (the threshold). More precisely, each of the values β_i, computed in round A, is correct (i.e., it is equal to the value of the function, which is 1 iff $\ell \geq k$) with probability which is larger than 1/2 by at least $\Omega(\sqrt{m_1/n})$.

Lemma 3.3 *Let $\ell = \sum_i b_i$ and $p < 1/2$ be a constant. For some constant c_1:*
(1) if $\ell \geq k$ then $Prob[A_i > \theta_k] > 1/2 + c_1\sqrt{m_1/n}$; and
(2) if $\ell < k$ then $Prob[A_i < \theta_k] > 1/2 + c_1\sqrt{m_1/n}$.

Proof: By Lemma 3.2, the expected value of A_i is $\mu_\ell = pnm_2 + 2m_1\ell$. For $k \leq \ell$, we have $\theta_k < \mu_k \leq \mu_\ell$, and moreover $\mu_\ell - \theta_k \geq m_1$. By Lemma 2.2,

$$(3) \qquad Pr[A_i > \theta_k] \geq 1 - B(\theta_k; N, r_\ell),$$

where $N = m_2 n$ and $r_\ell = \mu_\ell/N$. (Note that $r_\ell = (1-\ell/n)p + \ell/n(1-p)$ is always between p and $1-p$.) Unfortunately, it seems that using a standard Chernoff bound (e.g. Lemma 2.1) would not give us much. On the other hand, it is clear that $Pr[A_i > \theta_k]$ is minimized when the difference between k and ℓ is minimized, therefore we can assume, without loss of generality, that $\ell = k$. In addition, we need a refined analysis that bounds the individual binomial coefficients. By Lemma 2.3, the median for the binomial distribution is either $\lfloor \mu_\ell \rfloor$ or $\lceil \mu_\ell \rceil$. We can bound the desired probability,

$$(4)\ B(\theta_k; N, r_k) \leq 1/2 - \sum_{j=0}^{\lceil \mu_k \rceil - \lceil \theta_k \rceil} b(\lceil \mu_k \rceil - j; N, r_k).$$

Note that $\lceil \mu_k \rceil - \lceil \theta_k \rceil \geq m_1$. By applying Lemma 2.4, for each $0 \leq j \leq m_1$,

$$(5) \qquad b(\mu_k - j; N, r_k) = \Theta(\frac{1}{\sqrt{N}}).$$

Therefore, by substituting (4) and (5) into (3) and by the choice of parameters ($N = m_2 n$ and $m_2 = \Theta(m_1)$) we get

$$\begin{aligned} Pr[A_i > \theta_k] &\geq 1 - \left(\frac{1}{2} - m_1 \cdot \Theta(\frac{1}{\sqrt{m_2 n}})\right) \\ &= \frac{1}{2} + c_1\sqrt{m_1/n}, \end{aligned}$$

which completes the proof for the case that $k \leq \ell$. The case where $k > \ell$ is similar. □

Lemma 3.4 *At the end of round A, with probability at least* $1 - e^{-2c_2^2 m_1}$, *at least* $n/2 + c_2\sqrt{m_1 n}$ *of the values* β_i *are correct, where* $c_2 = c_1/2$.

Proof: By Lemma 3.3, for each processor P_i, the probability that we set β_i correctly is $1/2 + c_1\sqrt{m_1/n}$. Consider the probability that the number of correct β_i's is smaller than $n/2 + c_2\sqrt{nm_1}$. By Lemma 2.1, this probability is at most $e^{-2(c_2\sqrt{m_1/n})^2 n} = e^{-2c_2^2 m_1}$. □

Lemma 3.5 *Assume that at the beginning of round B there are at least* $n/2 + c_2\sqrt{nm_1}$ *of the* β_i*'s with the correct value. Then, at the end of round B, with probability* $1 - e^{-2q^2 n}$, *at least* $(1-2q)n$ *of the processors have the correct value, where* $q = e^{-2(c_2(1-2p))^2 m_1} < e^{-2}$, *and* $m_1 > (c_2(1-2p))^{-2}$.

Proof: Each γ_i is a random variable which is the majority of n random variables, β_i, corrupted by noise rate p. By the assumption of the lemma, at least $n/2 + c_2\sqrt{nm_1}$ of the β_i's are correct and at most $n/2 - c_2\sqrt{nm_1}$ are incorrect. We want to bound the probability that a processor, after the noise, gets more incorrect values than correct. Using Lemma 2.2 we can bound this event by $B(n/2; n, r)$, where

$$\begin{aligned} r &= \left(\frac{1}{2} + c_2\sqrt{\frac{m_1}{n}}\right)\cdot(1-p) + \left(\frac{1}{2} - c_2\sqrt{\frac{m_1}{n}}\right)\cdot p \\ &= \frac{1}{2} + c_2(1-2p)\sqrt{\frac{m_1}{n}}. \end{aligned}$$

Using Lemma 2.1 we can bound this probability by $q = e^{-2(c_2(1-2p)\sqrt{m_1/n})^2 n} = e^{-2(c_2(1-2p))^2 m_1}$. Therefore, each γ_i has a probability at least $1-q$ of being correct. By the choice of m_1 we ensure that $q < e^{-2}$.

We need to bound the probability that the number of correct γ_i's is significantly smaller than $(1-q)n$ (which is a lower bound on the expected number of correct γ_i's). Using again Lemma 2.1, with probability $1 - e^{-2q^2 n}$, at least $(1-2q)n$ of the γ_i's have the correct value. □

Lemma 3.6 *Given that at the beginning of round C at least* $(1-2q)n$ *(for* $q < e^{-2}$*) of the bits* γ_i *have the correct value, then at the end of round C, with probability* $1 - ne^{-2((1/2-2q)(1-2p))^2 n}$, *all the processors output the correct value.*

Proof: By Lemma 2.2, we can bound the probability that a given output is incorrect by $B(n/2; n, r)$, where r is the average probability of success. That is,

$$r = (1-2q)(1-p) + 2qp = \frac{1}{2} + (\frac{1}{2} - 2q)(1-2p).$$

Using Lemma 2.1, we bound this probability by at most $e^{-2((1/2-2q)(1-2p))^2 n}$, for each processor. Summing all the processors we get the desired bound. □

We can now establish the correctness of the procedure, with respect to the accuracy parameter ϵ.

Theorem 3.7 *Let* $\ell = \sum_{i=1}^{n} b_i$. *For any* k *and* ϵ, *and sufficiently large* n, *the procedure* `THRESHOLD`(k, ϵ), *with probability* $1 - \epsilon$ *if* $\ell \geq k$, *each processor* P_i *has* `output`$_i = 1$, *and if* $\ell < k$, *each processor* P_i *has* `output`$_i = 0$.

Proof: By Lemma 3.4 after round A, with probability at least $1 - \epsilon/3$ we have that at least $n/2 + c_2\sqrt{m_1 n}$ of the values of β_i are correct. (This is since $m_1 \geq (\ln(3/\epsilon))/(2c_2^2)$.)

By Lemma 3.5, if the protocol does not fail in round A then, with probability at least $1 - e^{-2ne^{-2}}$, we have at least $(1 - 2e^{-2})n$ with the correct value. Note that the error probability in this round is going to zero as n increases, and therefore, for sufficiently large n the failure probability is less than $\epsilon/3$.

By Lemma 3.6, if the protocol does not fail in rounds A and B then, with probability at least $1 - ne^{-2(1/2-2e^{-2})^2 n}$, all the processors output the correct value. Again, if we chose n sufficiently large, this will be less than $\epsilon/3$. Hence the total failure probability is bounded by ϵ.

The parameter m_1 has two constraints, which are independent of n. The first appears in Lemma 3.5, which requires $m_1 > (c_2(1-2p))^{-2}$. The second, from Lemma 3.4, requires that $m_1 \geq (\ln(3/\epsilon))/(2c_2^2)$. Our choice of m_1 meets both bounds. □

4 Unknown Noise Rate (p)

In this section we present a protocol for computing threshold functions even in the case where the noise rate is unknown. A simple way to try transforming the algorithm of the previous section to the case where the noise rate is unknown is to first estimate the noise rate and then use this estimate, $\hat{p}$, in place of the real noise rate, p. When considering the algorithm one can observe that an error of $O(1/n)$ in $\hat{p}$ would increase θ_k by only a constant and hence will not matter. Unfortunately, the difference between p and $\hat{p}$ would be much larger (in all estimation methods that we know of, that use only $O(n)$ broadcasts); this would cause this approach to fail. (For example, assume that initially each processor sends a zero bit, and then the number of ones received by each processor serves as an estimate on the error rate. For a single processor, the expected error would be $\Theta(1/\sqrt{n})$, which is much too large. If we could combine together

the error estimates computed at all the processors we would have gotten a much better estimate. But we do not know how to compute this with $O(n)$ broadcasts.)

Here we take a different approach to overcome the unknown noise rate. In fact, we never attempt to estimate the noise rate p, we only assume that we are given some upper bound p' on it, i.e., $p < p' < 1/2$. The idea is to replace the constant θ_k by a random variable Θ_k, such that $E[\Theta_k] = \theta_k$ and to reprove Lemma 3.3 with Θ_k instead of θ_k. (We need to get a non-trivial bound on the noise, p', since a direct estimate of the noise rate would incur an error of $\Omega(1/\sqrt{n})$, as explained above. Also our running time would depend on the distance of the bound p' from 1/2.)

First, we define the following round of broadcast. Each of the processors $P_1, \ldots, P_k$ sends a bit 1, and each of the processors $P_{k+1}, \ldots, P_n$, sends a bit 0. Each processor repeats this m_2 times, where m_2 is defined using p' instead of p, i.e., $m_2 = 2m_1/(1-2p')$ and $m_1 = (2 + \ln 1/\epsilon)/(c_2(1-2p'))^2$. Processor P_i receives $m_2 n$ bits $\eta_{i,j}$ and sets $\Theta_{k,i} = \sum_{j=1}^{m_2 n} \eta_{i,j} - m_1$ (each processor P_i has its own value $\Theta_{k,i}$).

Note that the $\Theta_{k,i}$ are independently and identically distributed. In addition, the expectation of $\Theta_{k,i}$ is,

$$\begin{aligned} E[\Theta_{k,i}] &= [(1-p)m_2 k + p m_2 (n-k)] - m_1 \\ &= m_2 pn + m_2 k(1-2p) - m_1 \\ &= \theta_k \ . \end{aligned}$$

It remains to show a lemma analogous to Lemma 3.3.

Lemma 4.1 *Let $\ell = \sum_i b_i$ and $p < 1/2$ a constant. For some constant c_3:*
(1) if $\ell \geq k$ then $Prob[A_i > \Theta_{k,i}] > 1/2 + c_3\sqrt{m_1/n}$; and
(2) if $\ell < k$ then $Prob[A_i < \Theta_{k,i}] > 1/2 + c_3\sqrt{m_1/n}$.

Proof: First consider the case $\ell \geq k$. Clearly the probability that A_i is larger than $\Theta_{k,i}$ is minimized when $\ell = k$. In this case the random variables A_i and $\Theta_{k,i} + m_1$ are identical. To see this, we can match processors $P_1, \ldots, P_k$, in the computation of $\Theta_{k,i}$, to the k processors that have input 1, in the computation of A_i. Similarly we can match processors $P_{k+1}, \ldots, P_n$, in the computation of $\Theta_{k,i}$, to the remaining $n-k$ processors with input 0, in the computation of A_i. Therefore, the random variables A_i and $\Theta_{k,i} + m_1$ are identical, and we conclude that,

$$Pr[A_i \geq \Theta_{k,i} + m_1] \geq \frac{1}{2}.$$

For the proof we will use the following identity,

$$\begin{aligned} Pr[A_i > \Theta_{k,i}] &= Pr[A_i \geq \Theta_{k,i} + m_1] \\ &+ Pr[\Theta_{k,i} + m_1 > A_i > \Theta_{k,i}]. \end{aligned}$$

The first term is at least half, as explained above, so now we are interested in bounding the second term, which will give the bias that A_i has over $\Theta_{k,i}$.

Consider the sequence of pairs of α's and η's from the matched processors. For a pair of matched processors, a and b, both $\alpha_{i,a}$ and the $\eta_{i,b}$ are distributed identically and independently. Therefore, given that $\alpha_{i,a} \neq \eta_{i,b}$, the probability that $1 = \alpha_{i,a} > \eta_{i,b} = 0$ is exactly half. Also, the probability that $\alpha_{i,a} \neq \eta_{i,b}$ is $2p(1-p)$. This means that the number of pairs of α and η such that $\alpha_{i,a} \neq \eta_{i,b}$, denoted t, has expectation $2p(1-p)m_2 n$. Using Lemma 2.1, with probability $1 - e^{-(p(1-p))^2 m_2 n} = 1 - e^{-c'n}$, we have $t \in [p(1-p)m_2 n, 3p(1-p)m_2 n]$.

Let the difference between the sum of the α's and the sum of η's be Δ, i.e. $\Delta = \sum_{i,j} \alpha_{i,j} - \sum_{i,j} \eta_{i,j}$. Given that there are exactly t places where the α's and η's differ, the probability that $\Delta \leq m_1$ is $O(m_1/\sqrt{t})$. Assuming that t is in the range $[p(1-p)m_2 n, 3p(1-p)m_2 n]$, we have that the probability is at least

$$c_4\sqrt{\frac{(1-2p')m_1}{p(1-p)n}} = c_5\sqrt{\frac{m_1}{n}},$$

for a constant p. Therefore,

$$\begin{aligned} Pr[A_i > \Theta_{k,i}] &= Pr[A_i \geq \Theta_{k,i} + m_1] \\ &+ Pr[\Theta_{k,i} + m_1 > A_i > \Theta_{k,i}] \\ &> \frac{1}{2} + c_5\sqrt{\frac{m_1}{n}} - e^{-c'n} \\ &= \frac{1}{2} + c_3\sqrt{\frac{m_1}{n}} \end{aligned}$$

This concludes the case that $\ell \geq k$.

Now, assume that $\ell < k$. Again, the probability would be minimized when $\ell = k - 1$. In this case we can match $n-1$ processors, and be left with a pair of processors one which sends 1 (in $\Theta_{k,i}$) and the other sends 0 (in A_i). Let the absolute difference between the sum of the η's and the α's in the $n-1$ matched processors be Δ, i.e. $\Delta = |(\sum \eta_{i,j}) - (\sum \alpha_{i,j})|$. We will bound our event by using the following identity,

$$\begin{aligned} Pr[A_i < \Theta_{k,i}] &= Pr[A_i < \Theta_{k,i} | \Delta \leq m_1] \cdot Pr[\Delta \leq m_1] \\ &+ Pr[A_i < \Theta_{k,i} | \Delta > m_1] \cdot Pr[\Delta > m_1] \end{aligned}$$

where $m_1 = (1-2p')m_2/2$. Intuitively, when Δ is less than m_1, the pair of unmatched processors "decides" the outcome.

Let t be the number of places in which the α's and η's differ. Then, the probability that $\Delta \leq m_1$ is $c_6 m_1/\sqrt{t}$, for some constant c_6. Again, with probability

$1-e^{-c'n}$ we have that $t \in [m_2 np(1-p), 4m_2 np(1-p)]$, therefore,

$$Pr[\Delta \leq m_1] \geq c_6 \frac{m_1}{\sqrt{t}} - e^{-c'n} = c_7 \sqrt{\frac{m_1}{n}} = q$$

for a constant p.

Now consider the pair of unmatched processors. The expected difference in the unmatched processors is at least $(1-2p')m_2 = 2m_1$. Therefore, the difference in the unmatched processors is more than m_1 with probability at least $1-e^{-2(1-2p')^2 m_2} = 1-e^{-4(1-2p')m_1} > 2/3$, for $m_1 > \ln 3/4(1-2p')$. This implies that, given that $\Delta \leq m_1$, the probability that $\Theta_{k,i} > A_i$ is strictly bounded away from half. Formally,

$$Pr[\Theta_{k,i} > A_i | \Delta \leq m_1] \geq \frac{2}{3} .$$

Given that $\Delta > m_1$, it is equally likely to favor A_i or $\Theta_{k,i}$. Since the bias in the unmatched pair is in favor of $\Theta_{k,i}$ we have,

$$Pr[\Theta_{k,i} > A_i | \Delta \leq m_1] \geq 1/2$$

To conclude,

$$\begin{aligned} & Pr[A_i < \Theta_{k,i}] \\ = \; & Pr[A_i < \Theta_{k,i} | \Delta \leq m_1] \cdot Pr[\Delta \leq m_1] \\ + \; & Pr[A_i < \Theta_{k,i} | \Delta > m_1] \cdot Pr[\Delta > m_1] \\ \geq \; & \frac{2}{3} q + \frac{1}{2}(1-q) = \frac{1}{2} + \frac{q}{6}. \end{aligned}$$

This concludes the case $k > \ell$. □

Once we established Lemma 4.1, the proof is the same as before. Therefore we established the following theorem.

Theorem 4.2 *There exists a protocol with $O(n)$ broadcasts that computes any threshold function, even if the noise rate is unknown.*

References

[ABLP91] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. "A Lower Bound for Radio Broadcast", *Journal of Computer and System Sciences*, 43:290–298, 1991.

[ABLP92] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. "Single Round Simulation on Radio Networks", *Journal of Algorithms*, 13:188–210, 1992.

[BGI92] R. Bar-Yehuda, O. Goldreich, and A. Itai. "On the Time-Complexity of Broadcast in Multi-Hop Radio Networks: An Exponential Gap between Determinism and Randomization" *Journal of Computer and System Sciences*, 45:104–126, 1992.

[Che52] H. Chernoff, "A Measure of the Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations", *Annals of Mathematical Statistics*, Vol. 23, pp. 493-509, 1952.

[CK85] I. Chlamtac and S. Kutten, "On Broadcasting in Radio Networks – Problem Analysis and Protocol Design", *IEEE Trans. Comm.*, COM-33, 1985.

[CK87] I. Chlamtac and S. Kutten, "Tree-Based Broadcasting in Multihop Radio Networks", *IEEECOM*, C-36:1209–1223, 1987.

[CW87] I. Chlamtac and O. Weinstein, "The Wave Expansion Approach to Broadcasting in Multihop Radio Networks", *INFOCOM*, pages 874–881, 1987.

[GM95] I. Gaber and Y. Mansour, "Broadcast in Radio Networks", *Proc. of 6th SODA*, pages 577–585, 1995.

[KM93] E. Kushilevitz, and Y. Mansour, "An $\Omega(D \log(N/D))$ Lower Bound for Broadcast in Radio Networks", *Proc. of 12th PODC*, pages 65-74, 1993. To appear in *SIAM J. on Computing.*

[DO77a] R. L. Dobrushin, and S. I. Ortyukov, "Lower Bound for the Redundancy of Self-Correcting Arrangements of Unreliable Functions", *Problems of Information Transmission*, Vol. 13, pp. 59–65, 1977.

[DO77b] R. L. Dobrushin, and S. I. Ortyukov, "Upper Bound for the Redundancy of Self-Correcting Arrangements of Unreliable Functions", *Problems of Information Transmission*, Vol. 13, pp. 203–218, 1977.

[EP96] W. Evans, and N. Pippenger, "Lower Bounds for Noisy Boolean Decision Trees", *Proc. of 28th STOC*, pp. 620-628, 1996.

[FPRU90] U. Feige, D. Peleg, P. Raghavan, and E. Upfal, "Computing with Unreliable Information", *Proc. of 22nd STOC*, pp. 128–137, 1990.

[Gác86] P. Gács, "Reliable Computation with Cellular Automata", *JCSS*, Vol. 32, pp. 15–78, 1986.

[Gál91] A. Gál, "Lower Bounds for the Complexity of Reliable Boolean Circuits with Noisy Gate", *Proc. of 32nd FOCS*, pp. 594-601, 1991.

[Gal88] R. G. Gallager, "Finding Parity in a Simple Broadcast Network", *IEEE Transactions on Information Theory*, Vol. 34, pp. 176-180, 1988.

[Hof56] W. Hoeffding, "On the Distribution of the Number of Successes in Independent Trials", *Annals of Mathematical Statistics*, Vol. 27, pp. 713-721, 1956.

[JS68] K. Jogdeg, and S. M. Samuels, "Monotone Convergence of Binomial Probabilities and a Generalization of Ramanujan Equation", *Annals of Mathematical Statistics*, Vol. 39, pp. 1191-1195, 1968.

[Kuz73] A. V. Kuznetsov, "Information Storage in a Memory Assembled from Unreliable Components", *Problems of Information Transmission*, Vol. 9, pp. 254–264, 1973.

[N56] J. von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components" *Automated Studies*, C. E. Shannon and J. McCarthy Eds., Princeton University Press, pp. 329-378, 1956.

[Pip85] N. Pippenger, "On Networks of Noisy Gates", *Proc. of 26th FOCS*, pp. 30-36, 1985.

[RS91] R. Reischuk, and B. Schmeltz, "Reliable Computation with Noisy Circuits and Decision Trees – A General $n \log n$ Lower Bound", *Proc. of 32nd FOCS*, pp. 602-611, 1991.

[RS94] S. Rajagopalan, and L. J. Schulman, "A Coding Theorem for Distributed Computing", *Proc. of 26th STOC*, pp. 790-799, 1994.

[Sch92] L. J. Schulman, "Communication on Noisy Channels: A Coding Theorem for Computation", *Proc. of 33rd FOCS*, pp. 724-733, 1992.

[Sch93] L. J. Schulman, "Deterministic Coding for Interactive Communication", *Proc. of 25th STOC*, pp. 747-756, 1993.

[SF96] R. Sedgewick and P. Flajolet, "An Introduction to the Analysis of Algorithms", Addison-Wesley, 1996.

[Spi96] D. A. Spielman, "Highly Fault-Tolerant Parallel Computation", *Proc. of 37th FOCS*, pp. 154-163, 1996.

[Tan81] A. S. Tanenbaum. *Computer Networks.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[Tay68] M. G. Taylor, "Reliable Information Storage in Memories Designed from Unreliable Components", *Bell Systems Technical Journal*, vol. 47, pp. 2299–2337, 1968.

[Yao97] A. C. Yao, "On the Complexity of Communication under Noise", invited talk in the *5th ISTCS conference*, 1997.

A Proof of Lemma 2.4

By definition,

$$b(k;n,p) = \frac{k+1}{n-k} \cdot \frac{1-p}{p} \cdot b(k+1;n,p) \quad .$$

For $k = \lfloor pn \rfloor - j$ we get

$$\begin{aligned} & b(\lfloor pn \rfloor - j; n, p) \\ = & \frac{\lfloor pn \rfloor - j + 1}{n - \lfloor pn \rfloor + j} \cdot \frac{1-p}{p} \cdot b(\lfloor pn \rfloor - j + 1; n, p) \\ \leq & \frac{pn - j + 1 - p\lfloor pn \rfloor + pj - p}{np - p\lfloor pn \rfloor + pj} \cdot b(\lfloor pn \rfloor - j + 1; n, p) \\ = & (1 - \frac{j+p-1}{np - p\lfloor pn \rfloor + pj}) \cdot b(\lfloor pn \rfloor - j; n, p) \quad . \end{aligned}$$

Also, using Stirling formula (see Equation (1)),

$$b(\lfloor pn \rfloor ; n, p) = \frac{1 + O(1/n)}{\sqrt{2\pi p(1-p)n}} \quad .$$

Therefore, by applying the above (and Equation (2)) we get:

$$\begin{aligned} & b(\lfloor pn \rfloor - c; n, p) \\ = & (\prod_{j=1}^{c} (1 - \frac{j+p-1}{np - p\lfloor pn \rfloor + pj})) \cdot b(\lfloor pn \rfloor ; n, p) \\ = & (1 - \frac{\Theta(c^2)}{2p(1-p)n} + O(\frac{1}{n^2})) \frac{1 + O(1/n)}{\sqrt{2\pi p(1-p)n}} \\ = & \Theta(\frac{1}{\sqrt{n}}), \end{aligned}$$

for constants p and c.