A Tight Lower Bound for Parity in Noisy Communication Networks

Chinmoy Dutta* Yashodhan Kanoria† D. Manjunath‡ Jaikumar Radhakrishnan*

Abstract

We show a tight lower bound of $\Omega(N \log\log N)$ on the number of transmission required to compute the parity of N bits (with constant error) in a network of N randomly placed sensors, communicating using local transmissions, and operating with power near the connectivity threshold. This result settles a question left open by Ying, Srikant and Dullerud (WiOpt 06), who showed how the sum of all N bits can be computed using $O(N \log\log N)$ transmissions.

Earlier works on lower bounds for communication networks worked with the full broadcast model without using the fact that the communication in real networks is local, determined by the power of the transmitters. In fact, in full broadcast networks parity can be computed using $O(N)$ transmissions. To obtain our lower bound we employ techniques developed by Goyal, Kindler and Saks (FOCS 05), who showed lower bounds in the full broadcast model by reducing the problem to a model of noisy decision trees. However, in order to capture the limited range of transmissions in real sensor networks, we define and work with a localized version of noisy decision trees. Our lower bound is obtained by exploiting special properties of parity computations in such decision trees.

1 Introduction

Since inexpensive wireless technology and sensing hardware are expected to be widely available and used, much recent effort has been devoted to developing models for these networks and protocols based on these models. A wireless sensor network consists of sensors that collect and cooperatively process data in order to compute some global function. The sensors interact with each other by transmitting wireless messages based on some protocol. The protocol is required to tolerate errors in transmissions since wireless messages typically are noisy.

In the problem we study, each sensor is required to detect a bit; then, all the sensors are required to collectively compute the parity of these bits. The difficulty of this task, of course, depends on the noise and the connectivity of the network. In this paper, we assume that each bit sent is flipped (independently for each receiver) with probability $\epsilon > 0$ during transmission. As for connectivity, we adopt the widely used model of random planar networks. Here the sensors are assumed to be randomly placed in a unit square. Then each transmission is assumed to be received (with noise) by the sensors that are within some prescribed radius of the sender. The radius is determined by the amount of power used by the sensors, and naturally one wishes to keep the power used as low as possible, perhaps just enough to ensure that the entire network is connected. It has been shown by Gupta and Kumar [4] that the threshold of connectivity is $\theta\left(\sqrt{\frac{\ln n}{n}}\right)$ (with a radius much smaller than this the network will not be connected almost surely, and with radius much larger it will be connected almost surely).

Our work is motivated by a protocol presented by Ying, Srikant and Dullerud [9] for computing the sum of all the bits (and hence any symmetric functions of these bits). They showed that even with the radius of transmission just near the connectivity threshold, and constant noise probability, one can compute the sum using a total of $O(n \log\log n)$ transmissions. They observed the (trivial) lower bound of n transmissions (for every processors must send at least one message), but left open the possibility of better upper bounds. One can compute the parity of the input bits from their sum; in fact, Ying et al. suggested that parity computation might be significantly easier than computing the sum. In this work, we prove a lower bound showing that the protocol of Ying et al. is optimal up to constant factors for computing the parity of the input bits. In order to state our result formally we need to define the model of noisy communication networks.

Definition 1.1. (Noisy network, protocol) *A communication network is an undirected graph G whose vertices correspond to processors and edges correspond to communication links. A message sent by a processor is received by all its neighbors.*

Noise: *In an ϵ-noise network, the messages are subjected to noise as follows. Suppose processor v sends bit b in time step t. Each neighbor of v then receives an independent noisy version of b; that is, the neighbor w of v receives the bit $b \oplus \eta_{w,t}$, where $\eta_{w,t}$ is an ϵ-noisy bit (that takes the value 1 with probability ϵ and 0 with probability $1-\epsilon$), these*

[*] Tata Institute of Fundamental Research, Mumbai, INDIA. email: {chinmoy,jaikumar}@tifr.res.in

[†] Stanford University, USA. email: ykanoria@stanford.edu The work was done while this author was at Indian Institute of Technology, Mumbai, INDIA.

[‡] Indian Institute of Technology, Mumbai, INDIA. email: dmanju@ee.iitb.ac.in

noisy bits being mutually independent.

Input: *An input to the network is an assignment of bits to the processors, and is formally an element of* $\{0,1\}^{V(G)}$.

Protocol: *A protocol on G for computing a function* $f : \{0,1\}^{V(G)} \rightarrow \{0,1\}$ *works as follows. The processors take turns to send single bit messages, which are received only by the neighbors of the sender. In the end, a designated processor* $v^* \in V(G)$ *declares the answer. The cost of the protocol is the total number of bits transmitted. A message sent by a processor is a function of the bits that it possesses until then. The protocol with cost* T *is thus specified by a sequence of vertices* $\langle v_1, v_2, \dots, v_T \rangle$ *and a sequence of* T *functions* $\langle g_1, g_2, \dots, g_T \rangle$*, where* $g_t : \{0,1\}^{j_t} \rightarrow \{0,1\}$ *and* j_t *is the number of bits received by* v_t *before time step* t *(plus one if* v_t *is an input processor). Furthermore,* $v_T = v^*$*, and the final answer is obtained by computing* g_T*. Note that in our model the number of transmissions is the same for all inputs.*

Error: *Such a protocol is said to be a* δ*-error protocol, if for all inputs* $x \in \{0,1\}^{V(G)}$, $\Pr[\mathsf{output} = f(x)] \geq 1 - \delta$.

In this paper, we consider networks that arise out of random placement of processors in the unit square.

Definition 1.2. (Random planar network) *A random planar network* $\mathcal{N}(N,R)$ *is a random variable whose values are undirected graphs. The distribution of the random variable depends on two parameters:* N*, the number of vertices, and* R*, the transmission radius. The vertex set of* $\mathcal{N}(N,R)$ *is* $\{P_1, P_2, \dots, P_N\}$*. The edges are determined as follows. First, these processors are independently placed at random, uniformly in the unit square* $[0,1]^2$*. Then,*

$$E(\mathcal{N}) = \{(P_i, P_j) : \mathsf{dist}(P_i, P_j) \leq R\}.$$

Theorem 1.1. (Lower bound for parity) *Let* $R \leq N^{-\beta}$ *for some* $\beta > 0$*. Let* $\delta < \frac{1}{2}$ *and* $\epsilon \in (0,1)$*. Then, with probability* $1 - o(1)$ *(over the placement of processors) every* δ*-error protocol on* $\mathcal{N}(N,R)$ *with* ϵ*-noise for computing the parity function* $\oplus : \{0,1\}^{V(\mathcal{N})} \rightarrow \{0,1\}$ *requires* $\Omega(N \log\log N)$ *transmissions.*

Remark 1.1. *This lower bound on parity, in fact, implies that the sum cannot be computed up to a constant additive error with* $o(N \log\log N)$ *transmissions. We conjecture that one cannot approximate the sum to within an additive error of* N^α *(for some* $\alpha > 0$*) using* $O(N)$ *transmissions.*

Remark 1.2. *Our definition of noise assumes that all transmissions are subject to noise with probability exactly* ϵ*. In the literature, other models of error have been considered. Some protocols work even in the weaker model where this probability is at most* ϵ*. Our lower bound applies also to this model.*

Remark 1.3. *We require only an upper bound on the transmission radius. However, the result is meaningful only when* $R = \Omega(\sqrt{\frac{\log N}{N}})$*, for otherwise, with high probability, the network is not connected and cannot be expected to compute any function that depends on all its inputs.*

1.1 Related work

The most commonly studied noisy communication model allows full broadcasts, that is, all sensors receive all messages (with independent noise). In this model, Gallager [2] considered the problem of collecting all the bits at one sensor, and showed how this could be done using $O(N \log\log N)$ transmissions; this implies the same upper bound for computing any function of the input bits. More recently, in a remarkable result, Goyal, Kindler and Saks [3] showed that Gallagher's protocol was the best possible for collecting all the bits. However, they do not present any boolean function for which $\Omega(N \log\log N)$ transmissions are required.

In the full broadcast model, protocols for computing specific functions have also been studied in the literature. Feige and Raghavan [1] presented a protocol with $O(N \log^* N)$ transmissions for computing the OR of N bits; this result was improved by Newman [8], who gave a protocol with $O(N)$ transmissions. For computing threshold functions Kushilevitz and Mansour [7] showed a protocol with $O(N)$ transmissions, assuming that all messages are subject to noise with probability exactly ϵ. Under the same assumption, Goyal, Kindler and Saks [3] showed that the sum of all the bits (and hence all symmetric functions) could be computed with $O(N)$ transmissions.

In this paper we are concerned with networks arising from random placement of sensors, where considerations of power impose stringent limits on the transmission radius. In this model, besides Ying, Srikant and Dullerud's [9] protocol for computing the sum mentioned above, the only other result we are aware of is a protocol of Kanoria and Manjunath [6] that uses $O(N)$ transmissions to compute the OR functions. However, no non-trivial lower bounds that apply specifically to sensor networks with limited transmission radius have appeared in the literature.

1.2 Techniques

We now present an overview of the proof technique used to derive our lower bound. As we explain in more detail in the next section, the proof has two parts. The first part is geometric. Since the transmission radius is small, it is possible to decompose the vertices of the communication network into clusters. The nodes in the interior of each cluster will continue to receive input, but those on the boundary will have their input fixed (arbitrarily) and thereby become auxiliary processors that still participate in the protocol by sending and receiving messages. This graph theoretic decomposition is based on routine arguments involving the distribution points chosen randomly and independently from the unit square. This decomposition allows us to view the protocol as a combination of several protocols acting independently on each cluster.

To view the protocol as a combination of separate protocols is not straightforward and we need to revisit the arguments used by Goyal, Kindler and Saks [3] to obtain their lower bounds. A key insight in their proof was that protocols in noisy communication networks could be translated into what they called generalized noisy decision trees. We adapt their argument to our setting. For us it is important to ensure that the decomposition of the network (which was the consequence of the limited transmission radius) is reflected in the noisy decision tree we construct. So, we define a notion of noisy decision trees appropriate for our setting, and show how efficient protocols on decomposed networks can be translated to such decision trees of small depth.

The argument this far was general and did not use the fact that the ultimate goal was to compute the parity function. The final part of our argument shows that we can rearrange the decision tree so that queries made to variables in the same cluster of the decomposition appear at adjacent levels of the tree. This part crucially depends on the fact that we are trying to compute the parity function. After the rearrangement, we can view the entire computation as a sequence of noisy decision tree computations, one for each cluster. We conclude that in order to have low overall error, the computation in each cluster must have vanishingly small error probability. At this stage we can directly apply a result of Goyal, Kindler and Saks, which states that any decision tree that computes the parity function with error $o(1)$ must have superlinear depth. This dependence of depth on error is strong enough to yield our lower bound. Proving lower bounds for functions other than parity using this approach would require developing of techniques that would eliminate the need for the rearrangement argument and remains an interesting open problem.

The interesting feature of this argument is that we work with appropriately defined decision trees instead of directly with the decomposed protocol. Once inputs of processors have been set, they become auxiliary. However, they continue to participate in the protocol. In particular, they receive transmissions from processors with inputs and can potentially aid error correction by providing additional reception diversity, which is crucially exploited in many of the upper bounds. So it is not true that our decomposition immediately breaks the protocol into independent subprotocols, operating separately on different clusters. Nevertheless, when we translate the decomposed protocol into our model of decision trees, we can view the computation of the entire decision tree as a combination of independent decision subtrees, operating separately on different clusters. This provides us the required product property, from which one easily deduces that each individual subtree must compute the parity within its cluster very accurately.

1.3 Organization of the paper In Section 2, we state two lemmas corresponding to the two parts of the argument, and derive the lower bound for parity. These lemmas are proved in later sections. The argument to show that networks with limited transmission radius can be decomposed appears in Section 4. In Section 3 we present our analysis of decomposed protocols. Most of that section is devoted to manipulations of noisy decision trees and the analysis of parity computations on these trees. The translation of noisy decomposed protocols into noisy decision trees appears in Section 5.

2 Proof of the lower bound for parity

In our proof, we will need to consider networks where some processors receive no input. We now introduce the terminology applicable in such situations.

Definition 2.1. (Input and auxiliary processors) *Let $G = (V, E)$ be a communication network. Some of the processors in V are designated as input processors; and the rest as auxiliary processors. We use I to refer to the set of input processors, and A for the set of auxiliary processors. An input to such a network is an element of $\{0,1\}^I$ and a protocol on such a network computes a function $f : \{0,1\}^I \rightarrow \{0,1\}$.*

Definition 2.2. (Network decomposition) *Let $G = (I \cup A, E)$ be a communication network. An (n, k)-decomposition of G is a partition of the vertex set of G of the form $I = I_1 \cup \cdots \cup I_k$ and $A = A_0 \cup A_1 \cup \cdots \cup A_k$ such that for $j = 1, 2, \ldots, k$,*

(P1) $|I_j| = n$, and

(P2) the neighborhood of I_j is contained in $I_j \cup A_j$.

The transmission noise parameter ϵ remains the same. A protocol Π on G is said be a (d, D)-bounded protocol *with respect to the decomposition $\langle A_0, (I_j, A_j) : j = 1, 2, \ldots, k\rangle$ if*

(P3) a processor in I_j makes at most d transmissions, and

(P4) all processors in $I_j \cup A_j$ put together make at most D transmissions.

Part I: This part of our argument is based on the observation that in a random planar network, processors are typically distributed uniformly over the entire area. By fixing the inputs of some of the processors, we can create 'buffer zones' of auxiliary processors so that the remaining processors now fall into large number of well-separated large clusters.

LEMMA 2.1. *Suppose $R \leq N^{-\beta}$, for some $\beta > 0$. Then, with probability $1 - o(1)$ the following holds: if*

> *there is a δ-error protocol on $\mathcal{N} = (N, R)$ for computing the parity function (on N bits) with T transmissions,*

then

> *there is an (n, k)-decomposition of $\mathcal{N}$ and a δ-error (d, D)-bounded protocol for parity (on nk bits) with respect to this decomposition, where $n = \Omega(NR^2)$, $k = \Omega(1/R^2)$, $d = O(T/N)$ and $D = O(TR^2)$.*

This lemma is proved in Section 4

Part II. In the second part of our argument, we analyze such bounded protocols for decomposed networks. Our analysis closely follows that of Goyal, Kindler and Saks [3]. For showing lower bounds on the number of transmissions in a noisy communication protocol, Goyal et al. translated such protocols into *generalized noisy decision (gnd) trees.* To state their result, and describe how it is used in our proof, we need a definition.

DEFINITION 2.3. (ADVANTAGE) *Let μ be a distribution on $\{0,1\}^n$. Let $f : \{0,1\}^n \to \{+1,-1\}$ and $\mathcal{A} : \{0,1\}^n \to C$, where C is some set. Then, the advantage of $\mathcal{A}$ for f under μ is given by*

$$\mathsf{adv}_{f,\mu}(\mathcal{A}) = \max_{a:C\to[-1,+1]} |\operatorname{E}[f(X)a(\mathcal{A}(X))]|,$$

where X has distribution μ. We will also use this notation even when $\mathcal{A}$ corresponds to a randomized algorithm, in which case, the expectation is computed over X as well as the independent internal random choices made by $\mathcal{A}$.

The result of Goyal et al. showed that the output of a gnd tree of small depth cannot have high advantage for the parity function. The following theorem states this result precisely. For a distribution μ on $\{0,1\}^n$, let $\alpha_\mu(n, D, \epsilon) \overset{\Delta}{=} \max_{\mathcal{T}} \mathsf{adv}_{\oplus,\mu}(\mathcal{T})$, where $\mathcal{T}$ ranges over all gnd trees with ϵ-noise and depth at most D and inputs in $\{0,1\}^n$. The formal definition of gnd trees appears in Section 3.2, but for the discussion in this section all we need to know is that a gnd tree has three parameters associated with it: the number of inputs, the depth, and the noise probability.

THEOREM 2.1. (GOYAL, KINDLER AND SAKS [3]) *Let μ be the distribution on $\{0,1\}^n$ defined by $\mu(0^n) = \frac{1}{2}$ and $\mu(e) = \frac{1}{2n}$ for all e of weight 1. Then*

$$(2.1)\ \alpha_\mu(n, D, \epsilon) \leq 1 - \exp\left(-O\left(\frac{D\log^2(1/\epsilon)}{\epsilon^2 n}\right)\right)$$

LEMMA 2.2. (PRODUCT LEMMA) *For all distributions μ on $\{0,1\}^n$ and (d, D)-bounded protocol Π on a network with (n, k)-decomposition and ϵ-noisy transmissions, we have $\mathsf{adv}_{\oplus,\mu^k}(\Pi) \leq \alpha_\mu(n, 3D, \epsilon^d)^k$.*

Proof of Theorem 1.1. Let μ be the distribution defined in Theorem 2.1. By combining Lemmas 2.1 and 2.2, we conclude that with probability $1 - o(1)$ the following is true: if $\mathcal{N}(N, R)$ has a δ-error protocol with T transmissions, then

$$1 - 2\delta \leq \alpha_\mu(n, 3D, \epsilon^d)^k,$$

where $n = \Omega(NR^2)$, $k = \Omega(1/R^2)$, $d = O(T/N)$ and $D = O(TR^2)$. Our assumption that $R \leq N^{-\beta}$ when combined with (2.1) implies that $T = \Omega(N \log\log N)$.

3 Analysis of decomposed protocols

In this section, we prove Lemma 2.2 by analyzing decomposed protocols for parity. We present our argument in a top-down fashion, by first assuming that decomposable protocols can be translated to suitably defined restricted noisy decision trees and completing the proof of Lemma 2.2 (in Section 3.1); detailed arguments justifying our assumption are presented in Sections 3.2 and 5.

3.1 Proof of Lemma 2.2

DEFINITION 3.1. (NOISY DECISION TREE) *An (n, k)-noisy (read-once) decision tree is a model for processing inputs in $(\{0,1\}^n)^k$. It consists of a balanced tree of depth k, whose internal nodes are labelled by a noisy function $g_v : \{0,1\}^n \to C_v$, where C_v is the set of children of v. By a noisy function we mean a function whose output depends on its input and some internal randomness that is independent for different noisy* *computations performed in the tree. Once an input $x = \langle x_1, x_2, \dots, x_k\rangle \in (\{0,1\}^n)^k$ is fixed, the (random) output of the tree is determined by the following natural computation. We start at the root, and when we arrive at an internal node v at level i, we determine the next vertex by evaluating $g_v(x_i)$. The (random) output of the tree on input $x \in (\{0,1\}^n)^k$ is the leaf reached in the end. Let $L(\mathcal{T})$ denote the set of leaves of $\mathcal{T}$.*

In our translation of a (d, D)-bounded protocol for an (n, k)-decomposed network into a noisy decision tree, we will ensure that the individual functions computed at the internal nodes of the trees cannot predict the parity function well.

Lemma 3.1. (Translation lemma) *For any (d, D)-bounded protocol Π on an (n, k)-decomposed network with ϵ-noisy transmissions, and a distribution μ on $\{0,1\}^n$, there is a noisy decision tree $\mathcal{T}$ for inputs in $(\{0,1\}^n)^k$ such that*

- $\mathsf{adv}_{\oplus,\mu^k}(\mathcal{T}) \geq \mathsf{adv}_{\oplus,\mu^k}(\Pi)$;
- $\mathsf{adv}_{\oplus,\mu}(g) \leq \alpha_\mu(n, 3D, \epsilon^d)$ *for every function g that appears in $\mathcal{T}$.*

This lemma will be proved in the following sections. In this section, we show how it can be used to prove Lemma 2.2. The main observation in this section is the following 'product property' for the advantage of decision trees.

Lemma 3.2. (Advantage of noisy decision trees) *Let $h : \{0,1\}^n \to \{+1,-1\}$. Suppose $\mathcal{T}$ is a noisy decision tree for computing $f : (\{0,1\}^n)^k \to \{+1,-1\}$ defined by $f(\langle x_1, x_2, \dots, x_k\rangle) = \prod_{i=1}^k h(x_i)$. Suppose, for each function $\mathcal{A}$ that appears in $\mathcal{T}$ we have $\mathsf{adv}_{h,\mu}(g) \leq \alpha$. Then, $\mathsf{adv}_{f,\mu^k}(\mathcal{T}) \leq \alpha^k$.*

Proof of Lemma 2.2. The lemma follows immediately by combining Lemmas 3.1 and 3.2.

It remains to prove Lemma 3.2. We will make use of the following proposition.

Proposition 3.1. *Let X be random variable taking values in $\{0,1\}^n$ with distribution μ. Then, for all $\mathcal{A} : \{0,1\}^n \to C$ and $a : C \to \mathbb{R}$, $|\operatorname{E}[f(X)a(\mathcal{A}(X))]| \leq \mathsf{adv}_{f,\mu}(\mathcal{A}) \cdot |a|$, where $|a| = \max_{c\in C} |a(c)|$.*

Proof of Lemma 3.2. Fix $b : L(\mathcal{T}) \to [-1, +1]$. Let X take values in $(\{0,1\}^n)^k$ with distribution μ^k. We wish to show that

$$|\operatorname{E}[f(X)\mathcal{T}(X)]| \leq \alpha^k.$$

Let the (random) sequence of vertices visited by the computation of $\mathcal{T}$ on input X be $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k, \mathbf{v}_{k+1}$. For $i = 1, 2, \dots, k$ and v in level i of the tree (at distance $i-1$ from the root) let

$$\alpha_i(v) = \operatorname{E}[h(X_i)h(X_{i+1})\cdots h(X_k)b(\mathbf{v}_{k+1}) \mid \mathbf{v}_i = v].$$

We will show by reverse induction on i that $|\alpha_i(v)| \leq \alpha^{k+1-i}$. The claim will then follow by taking i to be 1 and v to be the root of $\mathcal{T}$. For the base case, we have

$$\begin{aligned}\alpha_k(v) &= \operatorname{E}[h(X_k)b(\mathbf{v}_{k+1}) \mid \mathbf{v}_k = v] \\ &= \operatorname{E}[h(X_k)b(g_v(X_k))] \\ &\leq \mathsf{adv}_{h,\mu}(g_v) \leq \alpha.\end{aligned}$$

For the induction step assume that $i < k$ and that $|\alpha_{i+1}(w)| \leq \alpha^{k-i}$ for all vertices w in level $i+1$ of the tree (at distance i from the root). Then, for a vertex v in level i, we have

$$\begin{aligned}|\alpha_i(v)| &= |\operatorname{E}[h(X_i)h(X_{i+1})\cdots h(X_k)b(\mathbf{v}_{k+1}) \mid \mathbf{v}_i = v]| \\ &= |\operatorname{E}[h(X_i)\alpha_{i+1}(g_v(X_i))]| \\ &\leq \mathsf{adv}_{h,\mu}(g_v) \cdot \max_w |\alpha_{i+1}(w)| \\ &\leq \alpha^{k+1-i},\end{aligned}$$

where we used Proposition 3.1 to justify the second last inequality, and the induction hypothesis to justify the last inequality.

3.2 Tree rearrangement This and the next section will be devoted to the proof of Lemma 3.1, which we assumed in the proof of Lemma 2.2. As stated earlier, we will use the method of Goyal, Kindler and Saks [3] to translate a communication protocol into a noisy decision tree. Then, we will show how this noisy decision tree can be rearranged to ensure that the tree has the required read-once property required in Definition 3.1. We will first state in the form of a lemma, the result obtained by a direct application of the arguments in Goyal, Kindler and Saks [3]. Then, assuming this this lemma, we will prove Lemma 3.1 in this section. In the next section, we will review the arguments of Goyal, Kindler and Saks [3] and apply them to our setting.

Definition 3.2. (Ordered oblivious tree) *A decision tree for the set of inputs S^k is a balanced tree where each internal node v is labelled by a pair $\langle i_v, g_v\rangle$ where $i_v \in [k]$ and $g_v : S_{i_v} \to C_v$, where C_v is the set of children of v. Such a tree $\mathcal{T}$ computes a function from S^k to the set $L(\mathcal{T})$ of leaves of $\mathcal{T}$ as follows: on input $\langle x_1, x_2, \dots, x_n\rangle$, the computation starts at the root and determines the next vertex to visit after a vertex v by evaluating $g_v(x_{i_v})$; the leaf reached in the end is the result of the computation. If a vertex v is labelled* *by $i \in [k]$, then we say that the i-th input variable is queried at that vertex. We say that the tree is oblivious if the label i_v of a vertex v depends only on v's distance from the root. We say that such a tree is* ordered *if for all $j \in [k]$ all queries to the the j-th input appear at consecutive levels.*

Remark 3.1. *In the definition above, the functions g_v are deterministic. However, in order to model noisy communication networks we will need to allow functions computed at the internal nodes to have noise. Goyal, Kindler and Saks modelled this noise by allowing the functions additional variables, which were set according to some distribution independent of the input (but based on a noise parameter δ).*

Definition 3.3. (GKS-tree) *An (n, k, D, δ)-GKS tree $\hat{\mathcal{T}}$ consists of an oblivious decision tree $\mathcal{T}$ on inputs S^k where $S = \{0,1\}^n \times (\{0,1\}^n)^{|\Lambda|}$ (for some index set Λ), where each function g_v has a special form:*

$$g_v(x_{i_v}, \overline{z}_{i_v}) = g'_v(x_{i_v} \oplus z_{i_v,\lambda_v}),$$

for some $g'_v : \{0,1\}^n \to C_v$ and $\lambda_v \in \Lambda$. Each input is queried at most D times in the tree. The computation of $\hat{\mathcal{T}}$ proceeds as follows: on input $x \in (\{0,1\}^n)^k$, each $\overline{z}_{i,\lambda} \in \{0,1\}^n$ is chosen independently according to the binomial distribution $\mathcal{B}(n, \delta)$. Once the entire input $(\overline{x}, \overline{z}) \in S^k$ is determined, we compute $\mathcal{T}(\overline{x}, \overline{z})$ as in Definition 3.2 above.

Remark 3.2. *When $k = 1$ the trees defined in the above definition correspond to the* **generalized noisy decision (gnd) trees** *of Goyal, Kindler and Saks [3].*

We can now state the main connection between protocols and decision trees.

Lemma 3.3. *For any (d, D)-bounded protocol Π on an (n, k)-decomposed network with ϵ-noisy transmissions, and distribution μ on $(\{0,1\}^n)^k$, there is an $(n, k, 3D, \epsilon^d)$-GKS tree $\mathcal{T}$ such that $\mathsf{adv}_{\oplus,\mu}(\mathcal{T}) \geq \mathsf{adv}_{\oplus,\mu}(\Pi)$.*

We present the detailed proof of this lemma in Section 5. Note that this lemma does not guarantee that the resulting GKS tree is ordered. Our main observation in this section is that decision trees can be assumed to be ordered when the inputs come from a product distribution, and we wish to approximate the parity function. To show this we will describe a method for rearranging an arbitrary oblivious decision tree so that it becomes ordered. To state this formally, we need one more definition.

Definition 3.4. (Tree rearrangement) *Let $\mathcal{T}$ and $\mathcal{T}'$ be oblivious decision trees for the same set of inputs. We say that $\mathcal{T}'$ is a rearrangement of tree $\mathcal{T}$ if*

- *both trees query each variable the same number of times;*
- *the functions labelling vertices of $\mathcal{T}'$ also appear in $\mathcal{T}$ (up to obvious renaming of children); formally, for every vertex $\hat{v}$ in $\mathcal{T}'$ labelled $(i, \hat{g})$, there is a vertex v in $\mathcal{T}$ labelled (i, g) in T and a bijection $\pi : C_{\hat{v}} \to C_v$ such that $\forall x \in S_i : \hat{g}(x) = \pi(g(x))$.*

Lemma 3.4. (Ordering lemma) *Let $f : S^k \to \{+1, -1\}$ of the form $f(x_1, x_2, \ldots, x_k) = h(x_1)h(x_2)\cdots h(x_k)$. Let μ be a product distribution on S^k. Then every oblivious decision tree $\mathcal{T}$ can be rearranged to obtain an ordered oblivious decision tree $\hat{\mathcal{T}}$ such that $\mathsf{adv}_{f,\mu}(\hat{\mathcal{T}}) \geq \mathsf{adv}_{f,\mu}(\mathcal{T})$.*

This lemma will follow immediately from the following claim.

Lemma 3.5. (Move to root) *Let $f : S^k \to \{+1, -1\}$ be of the form $f(x_1, x_2, \ldots, x_n) = h(x_1)h(x_2)\cdots h(x_n)$, where $h : S \to \{+1, -1\}$. Let μ be a product distribution on S^k. Let $\mathcal{T}$ be an oblivious decision tree with inputs in S^k such that the input x_n is queried only at the level just above the leaves. Then, $\mathcal{T}$ can be rearranged to obtain a tree $\hat{\mathcal{T}}$ where*

1. *the input x_n is queried only at the root;*
2. *for all $j \neq n$, if x_j was queried at level r of $\mathcal{T}$, then x_j is queried at level $r + 1$ of $\hat{\mathcal{T}}$;*
3. *$\mathsf{adv}_{f,\mu}(\hat{\mathcal{T}}) \geq \mathsf{adv}_{f,\mu}(\mathcal{T})$.*

Proof of lemma 3.5. Let $X = \langle X_1, X_2, \ldots, X_k \rangle$ take values in S^k with distribution μ; since μ is a product distribution the X_i's are independent. Suppose $\mathcal{T}$ makes t queries to the input. Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{t+1}$ be the random sequence of vertices visited by the computation of $\mathcal{T}$ on input X. Fix $b : L(\mathcal{T}) \to [-1, +1]$ such that

$$\begin{aligned}\mathsf{adv}_{f,\mu}(\mathcal{T}) &= |\,\mathrm{E}[h(X_1)h(X_2)\cdots h(X_n)b(v_{t+1})]| \\ &= |\,\mathrm{E}[\mathrm{E}[h(X_1)\cdots h(X_n)b(g_{\mathbf{v}_t}(X_n)) \mid \mathbf{v}_t]]|.\end{aligned}$$

Since X_n is queried only at the end, $h(X_1)\ldots h(X_{n-1})$ and $b(g_{\mathbf{v}_t}(X_n))$ are independent given $\mathbf{v}_t$, so $\mathrm{E}[h(X_1)\ldots h(X_{n-1})h(X_n)b(g_{\mathbf{v}_t}(X_n)) \mid \mathbf{v}_t]$ $= \mathrm{E}[h(X_1)\ldots h(X_{n-1}) \mid \mathbf{v}_t] \cdot \mathrm{E}[h(X_n)b(g_{\mathbf{v}_t}(X_n)) \mid \mathbf{v}_t]$.

Let $\alpha(v) = \mathrm{E}[h(X_1)\ldots h(X_{n-1}) \mid \mathbf{v}_t = v]$ and $\beta(v) = \mathrm{E}[h(X_n)b(g_{\mathbf{v}_t}(X_n)) \mid \mathbf{v}_t = v]$. Let $v^* =$

$\arg\max \beta(v)$; thus, among the functions labelling vertices that query X_n (at level t), g_v^* has the best advantage in the tree for h under the distribution of X_n. It is thus natural to expect (and not hard to verify) that if we replace all queries to X_n by this query g_{v^*}, the overall advantage can only improve. Once this is done, the last query does not depend on the previous query, and can, therefore, be moved to the root. We now present the argument formally. We have,

$$\begin{aligned}(3.2)\qquad \mathsf{adv}_{f,\mu}(\mathcal{T}) &= |\operatorname{E}[\alpha(\mathbf{v}_t)\beta(\mathbf{v}_t)]| \\ &\leq \operatorname{E}[|\alpha(v_T)|]\cdot|\beta(v^*)|.\end{aligned}$$

We are now ready to describe the rearrangement of $\mathcal{T}$. Let $\mathcal{T}^-$ be the subtree of $\mathcal{T}$ consisting of the first t levels of vertices; thus vertices where X_n is queried in $\mathcal{T}$ become leaves in $\mathcal{T}^-$. We first make $|C_{v^*}|$ copies of $\mathcal{T}^-$; we refer to these copies by $\mathcal{T}_c^-$ $(c \in C_{v^*})$, and assume that the root of $\mathcal{T}_c^-$ is renamed c. In the new tree $\hat{\mathcal{T}}$, we have a root with label $\langle n, g_{v^*}\rangle$ which is connected to the subtrees $\mathcal{T}^-$. We claim that $\mathsf{adv}_{f,\mu}(\hat{\mathcal{T}}) \geq \mathsf{adv}_{f,\mu}(\mathcal{T})$. Indeed, consider the function $\hat{b} : L(\hat{\mathcal{T}}) \to [-1,+1]$ that takes the value $\mathsf{sign}(\alpha(v))b(c)$ on the leaf in $\mathcal{T}_c^-$ corresponding to $v \in L(\mathcal{T}^-)$. Then, we have

$$\begin{aligned}(3.3)\ \mathsf{adv}_{f,\mu}(\hat{\mathcal{T}}) &\geq |\operatorname{E}[h(X_1)h(X_2)\cdots h(X_n)\hat{b}(\hat{v}_T)]| \\ &= \operatorname{E}[|\alpha(\mathbf{v}_T)|]|\beta(v^*)|.\end{aligned}$$

Lemma 3.5 now follows by combining (3.2) and (3.3).

We are now ready to show how trees computing the parity function can be reordered, and prove Lemma 3.4. The argument essentially involves repeated application of Lemma 3.5 to place all queries made to a variable in adjacent levels. We state the argument formally by considering a carefully defined *minimal counterexample.*

Proof of Lemma 3.4. Fix an oblivious decision tree $\mathcal{T}$. Let the depth $\mathcal{T}$ be r. We say that there is an *alternation at level* $\ell \in \{3,\ldots,r\}$ of $\mathcal{T}$ if the variable queried at level ℓ is queried at a level before $\ell-1$ but not at level $\ell-1$. Clearly, a tree with no alternations is an ordered tree. Among all rearrangements of $\mathcal{T}$, let $\hat{\mathcal{T}}$ be such that

(P1) $\mathsf{adv}_{f,\mu}(\hat{\mathcal{T}}) \geq \mathsf{adv}_{f,\mu}(\mathcal{T})$;

(P2) among all $\hat{\mathcal{T}}$ satisfying (P1), $\hat{\mathcal{T}}$ has the fewest alternations;

(P3) among all $\hat{\mathcal{T}}$ satisfying (P1) and (P2), the last alternation in $\hat{\mathcal{T}}$ is farthest from the root.

We claim that $\hat{\mathcal{T}}$ has no alternations. Let us assume that $\hat{\mathcal{T}}$ has alternations and arrive at a contradiction. Let $\hat{\mathcal{T}}'$ be the tree obtained from $\hat{\mathcal{T}}$ by merging queries on adjacent levels into one *superquery.* That is, if there are j adjacent levels somewhere in the tree that query x_i, with two outcomes, then we replace these j levels by a single superquery with 2^j outcomes. Note that the number of alternations in $\hat{\mathcal{T}}'$ is the same as in $\hat{\mathcal{T}}$. Let r' be the number of queries in $\hat{\mathcal{T}}'$. We consider two cases:

$\mathcal{T}'$ does not have an alternation at level r': Let x_1 be the variables queried at level r'. By Lemma 3.5, we obtain a tree $\hat{\mathcal{T}}''$ where the superquery to $x1$ appears only at the root, and all other superqueries are shifted one level down. Now, however, if each superquery in $\hat{\mathcal{T}}''$ is replaced by its corresponding subtree of queries from $\hat{\mathcal{T}}$, then we obtain a rearrangement of $\hat{\mathcal{T}}$ satisfying (P1) and (P2), but with alternation at a level farther from the root, contradicting (P3).

$\mathcal{T}'$ has an alternation at level r': Suppose x_1 is queried at level r', and the previous query to x_1 is at level $r'' < r'$ (with no queries to x_1 in the levels $r''+1, r''+2, \ldots, r'-1$). Now, we apply Lemma 3.5 to the subtrees of $\mathcal{T}'$ rooted at level $r''+1$, thereby obtaining a rearrangement $\hat{\mathcal{T}}''$, where x_1 is now queried at levels $r''+1$ instead of at level r'. Clearly, the resulting tree $\hat{\mathcal{T}}''$ has fewer alternations than $\hat{\mathcal{T}}'$. Furthermore, if each superquery in $\hat{\mathcal{T}}''$ is replaced by its corresponding tree of queries from $\hat{\mathcal{T}}$, we obtain a rearrangement of $\hat{\mathcal{T}}$. It can be verified that this rearrangement has advantage at least no worse than $\hat{\mathcal{T}}$ but has fewer alternations—contradicting (P2).

Proof of Lemma 3.1. By combining Lemmas 3.3 and 3.4 we see that Π can be converted into a ordered localized $(n,k,3D,\epsilon^d)$-GKS tree. Since this tree is ordered all queries to any particular variable appear in consecutive levels. In our final tree we will combine all these queries into a single query. In particular, if there are $\ell \leq 3D$ levels that query (x_i, z_i), then we collapse them, so as to yield a single query with 2^ℓ outcomes. Note, however, that the result of this query depends not only on the real input in $x_i \in \{0,1\}^n$ but also on the noise variable z_i. In the final noisy decision tree $\mathcal{T}$, we regard this superquery $g(x_i)$ as a noisy function of the input x_i, with z_i providing the internal randomness for its computation. Since $g(x_i)$ was derived from a gnd-tree of depth $\ell \leq 3D$ with noise ϵ^d, it follows from the definition of $\alpha_\mu(n,3D,\epsilon^d)$ that $\mathsf{adv}_{\oplus,\mu}(g) \leq \alpha_\mu(n,3D,\epsilon^d)$

4 Decomposing random planar networks: proof of Lemma 2.1

The random placement of processors in the unit square typically arranges them uniformly. We will exploit this uniformity to obtain the required decomposition.

Lemma 4.1. (Chernoff bounds) *Let X be the sum of N independent identically distributed indicator random variables. Let $\mu = E[X]$. Then,* $\Pr[X \leq \frac{1}{2}\mu] \leq \exp(-0.15\mu)$.

Proof. Our claims follow immediately from the following version of the Chernoff bound due to Hoeffding [5]: if the random variable Z has binomial distribution $\mathcal{B}(N,p)$, then

$$\Pr[X \geq (p+\delta)N] \leq \left(\frac{p}{p+\delta}\right)^{(p+\delta)N}\left(\frac{1-p}{1-p-\delta}\right)^{(1-p-\delta)N}. \quad (4.4)$$

To derive our claim, we consider the random variable $Y = N - X$, and apply (4.4) with $p = 1 - \frac{\mu}{N}$ and $\delta = \frac{\mu}{2N}$, to obtain

$$\begin{aligned}\Pr[X \leq \frac{1}{2}\mu] &\leq \Pr[Y \geq (p+\delta)N] \\ &\leq \left(1 - \frac{\delta}{p+\delta}\right)^{(p+\delta)N}\left(\frac{1-p}{1-p-\delta}\right)^{(1-p-\delta)N} \\ &\leq \exp(-\delta N)\cdot 2^{\frac{\mu}{2}} \\ &\leq \exp\left(-\frac{1}{2}(1-\ln 2)\mu\right) \\ &\leq \exp(-0.15\mu).\end{aligned}$$

Proof of Lemma 3.2. We tessellate the unit square into $\lfloor 1/R \rfloor$ cells, each a square of side $\frac{1}{\lfloor 1/R \rfloor}$. We number the rows and columns of this tessellation using indices in $\{1, 2, \ldots, \lfloor 1/R \rfloor\}$, and refer to the cell in the i-th row and j-th column by c_{ij}. (Later, we use c_{ij} to refer to the set of processors that fall in this cell.) We thus have a total of $M = \lfloor 1/R \rfloor^2$ cells. The expected number of processors in any one cell is $\mu = N/M$. Since $R \geq \sqrt{10 \ln N/N}$, we have $\mu \geq 10 \ln N$, and by Lemma 4.1, the probability that there are fewer than $\mu/2$ processors in any one cell is is $o(\frac{1}{M})$. So, with probability $1 - o(1)$, all cells have at least at least $N/(2M)$ processors.

Now, let $\mathcal{S}_1 = \{c_{ij} : i = 1 \pmod 3 \text{ and } j = 1 \pmod 3\}$. Then, $|\mathcal{S}_1| \geq M/9$. For each $c \in \mathcal{S}_1$, let the neighborhood of c, denoted by $\Gamma(c)$, be the set of (at most nine) cells that are within distance R of c. Note that distinct cells in $\mathcal{S}_1$ have disjoint neighborhoods. If the total number of transmissions in the original protocol is T, then the average number of transmissions made from $\Gamma(c)$ as c ranges over $\mathcal{S}_1$ is at most $9T/M$. By Markov's inequality, for at least half the cells $c \in \mathcal{S}_1$ fewer than $18T/M$ transmissions are made from $\Gamma(c)$. Let $\mathcal{S}_2$ be the set of these cells; $|\mathcal{S}_2| \geq M/18$. For each cell $c \in \mathcal{S}_2$, we identify the set I_c of $\lceil N/(4M) \rceil$ processors that make fewest transmissions. We are now ready to describe the decomposition of the planar communication network.

The set of input processors will be $I = \bigcup_{c \in \mathcal{S}_2} I_c$. We fix the input of all processors not in $\mathcal{I}$ at 0, and treat them as auxiliary processors. The protocol continues to compute the parity of the inputs provided to processors in I. For $c \in \mathcal{S}_2$, let A_c be the set of auxiliary processors in the cells in $\Gamma(c)$. (We include those auxiliary processors that are in no $\Gamma(c)$ in any of the A_c arbitrarily.) We have thus obtained a decomposition $\langle I_c : c \in \mathcal{S}_2 \rangle \cup \langle A_c : c \in \mathcal{S}_2 \rangle$, such that

(a) the number of input classes in the decomposition is $k = |\mathcal{S}_2| \geq M/18$;

(b) each input class has $n = \lceil \mu/4 \rceil$ processors;

(c) The total number of transmissions made by all processors in $I_c \cup A_c$ is at most $D = 18T/M$;

(d) The total number of transmissions made by any one processor is I_c is at most $d = D/n = 72T/N$.

The original protocol now reduces to a δ-error (d, D)-bounded protocol for computing the parity function on an (n, k)-decomposed network, where $n \geq NR^2/4$, $k \geq \frac{1}{18} \lfloor 1/R \rfloor^2$, $d \leq 72T/N$ and $D \leq 18TR^2$.

5 Translation from communication protocol to decision trees

We will translate a protocol into a decision tree and prove Lemma 3.3. Our proof borrows heavily from a similar proof of Goyal, Kindler and Saks; as in their proof, we will use two intermediate communication models before we finally derive the decision tree. In this section, for brevity, we use the notation (n, k, d, D)-protocol to mean a (d, D)-bounded protocol on an (n, k)-decomposed network.

Definition 5.1. (Intermediate protocols) *The following two kinds of protocols are obtained by imposing restrictions on decomposed protocols of Definition 2.2.*

Semi-noisy protocol: *An ϵ-noise (n, k, d, D)-semi-noisy protocol differs from an (n, k, d, D)-protocol only in the following respects.*

(a) When it is the turn of an input processor to send a message, it sends its input bit, whose independent ϵ-noisy copies are then received by its neighbors.

(b) A transmission made by an auxiliary processor is not subject to any noise.

Noisy copy protocol: *An ϵ-noise (n, k, D)-noisy-copy protocol is an ϵ-noise $(n, k, 1, D)$-semi-noisy protocol; in other words, every input processor makes exactly one broadcast, so that each of its neighbors receives exactly one independent ϵ-noisy copy of this input.*

Remark 5.1. *In these special kinds of protocols, the messages sent by the input processors does not depend on the messages these processors receive. Thus, we may assume that the input processors make their transmissions in the beginning of the protocol an appropriate number of times, and after that the auxiliary processors interact according to a zero noise protocol.*

Lemma 5.1. (From broadcast to semi-noisy) *For every function $f : (\{0,1\}^n)^k \to \{+1, -1\}$, distribution μ on $(\{0,1\}^n)^k$ and every ϵ-noise (n, k, d, D)-protocol Π, there is an ϵ-noise $(n, k, d, 3D)$-semi-noisy protocol Π_1 such that $\mathsf{adv}_{f,\mu}(\Pi) \leq \mathsf{adv}_{f,\mu}(\Pi_1)$.*

Lemma 5.2. (From semi-noisy to noisy-copy) *For every function $f : (\{0,1\}^n)^k \to \{+1, -1\}$, distribution μ on $(\{0,1\}^n)^k$ and every ϵ-noise (n, k, d, D)-semi-noisy protocol Π_1, there is an ϵ^d-noise (n, k, D)-noisy-copy protocol Π_2 such that $\mathsf{adv}_{f,\mu}(\Pi_1) \leq \mathsf{adv}_{f,\mu}(\Pi_2)$.*

Lemma 5.3. (From noisy-copy to GKS tree) *For every function $f : (\{0,1\}^n)^k \to \{+1, -1\}$, distribution μ on $(\{0,1\}^n)^k$ and every ϵ-noise (n, k, D)-noisy-copy protocol Π_2, there is an (n, k, D, ϵ)-GKS tree $\mathcal{T}$ such that $\mathsf{adv}_{f,\mu}(\Pi_2) \leq \mathsf{adv}_{f,\mu}(\mathcal{T})$.*

Proof of Lemma 5.1. Fix an (n, k, d, D)-protocol Π on a graph G. We will construct an $(n, k, d, 3D)$-semi-noisy protocol Π_1 on a $G_1 = (V_1, E_1)$. The graph G_1 will contain G as a subgraph; however, all vertices inherited from G will correspond to auxiliary processors. In addition, for each input vertex v of G, we will have a new input vertex v' in G_1, which will be connected to v and its neighbors in G. Let $(I = \bigcup_{j=1}^k I_j, A = A_0 \cup \bigcup_{j=1}^k A_j)$, be the decomposition corresponding to Π. The decomposition corresponding to Π_1 will be $(I' = \bigcup_{j=1}^k I'_j, A' = A_0 \cup \bigcup_{j=1}^k A'_j)$, where $I'_j = \{v' : v \in I_j\}$ and $A'_j = A_j \cup I_j$. Suppose Π uses T transmissions.

Notation: For $i = 1, 2, \ldots, T$ and $v \in V(G)$, let $b_v[i]$ be the bit received by v when the i-th transmission is made; if v does not receive the i-th transmission, we define $b_v[i]$ to be 0.

The protocol Π_1 for simulating Π will operate in T stages, one for each transmission made by Π. The goal is to ensure that in the end each auxiliary processor v of G_1 constructs a sequence $b'_v \in \{0,1\}^T$, such that $\langle b'_v : v \in V(G)\rangle$ and $\langle b_v : v \in V(G)\rangle$ (of the protocol Π) have the same distribution, for every input in $(\{0,1\})^n$. This implies that the outputs of Π' and Π have the same distribution. Suppose the first $\ell - 1$ stages have been successfully simulated and $\langle b'_v[1, \ldots, \ell - 1] : v \in V(G)\rangle$ have been appropriately constructed. We now describe how stage ℓ is implemented and $\langle b'_v[\ell] : v \in V(G)\rangle$ are constructed. If the ℓ-th transmission in Π is made by an auxiliary processor v in G, then it will be simulated in Π_1 using one noiseless transmission from v; if the ℓ-th transmission is made by an input vertex v of G, then it will be simulated in Π_1 using two (noiseless) transmissions from v and one ϵ-noisy transmission from the corresponding (newly added) input vertex v'.

v **is an auxiliary vertex in** G**:** The auxiliary vertex v in G_1 operates exactly in the same fashion as in G, and sends a bit b, which is received without error by all its neighbors. Each neighbor $w \in V(G)$ of v independently sets its bit $b'_w[\ell]$ to be an ϵ-noisy copy of b.

v **is an input vertex in** G**:** The auxiliary vertex v in G_1 has all the information that the corresponding input vertex v in G would have had, except the input (which is now given to the new input vertex v') . So, v transmits (with no noise) two bits, b_0 and b_1, corresponding to the two possible input values that v' might have. Next, the input vertex v' transmits its input c; let c_w denote the ϵ-noisy version of c that the neighbor w receives. Each neighbor $w \in V(G)$ of v now acts as follows: if $b_0 = b_1$, then it sets $b'_w[\ell]$ to be an ϵ-noisy copy of b_0 (using its internal randomness); if $b_0 \neq b_1$, then it sets $b'_w[\ell]$ to b_{c_w}.

Proof of Lemma 5.2. Let Π_1 be an (n, k, d, D)-semi-noisy protocol. As remarked above, all input processors in a semi-noisy protocol can be assumed to make their transmissions right in the beginning, after which only the auxiliary processors operate. Thus, each auxiliary processor receives at most d independent ϵ-noisy copies of the input from each input processor in its neighborhood. The following lemma of Goyal, Kindler and Saks [3] shows that a processor can generate d independent ϵ-noisy copies of any input from one ϵ^d-noisy copy.

> Let t be an arbitrary integer, $\epsilon \in (0, 1/2)$ and $\gamma = \epsilon^t$. There is a randomized algorithm that

> takes as input a single bit b and outputs a sequence of t bits and has the property that if the input is a γ-noisy copy of 0 (respectively of 1), then the output is a sequence of independent ϵ-noisy copies of 0 (respectively of 1).

We modify the protocol Π_1 to Π_2 by requiring that each input processor make one ϵ^d-noisy transmission of its input bit. Each auxiliary processor on receiving such an input uses its internal private randomness to extract the required ϵ-noisy copies. Then onwards the protocol proceeds as before. We may now fix internal randomness used by the auxiliary processors in such a way that the advantage of the resulting protocol for the input distribution μ is at least as good as that of the original protocol. Thus, all processors use (deterministic) functions to compute the bit that they transmit.

Proof of Lemma 5.3. Let Π_2 be an (n, k, D)-noisy-copy protocol, with the underlying decomposition $(I = \bigcup_{j=1}^{k} I_j, A = A_0 \cup \bigcup_{j=1}^{k} A_j)$. We will now show how this protocol can be simulated using a GKS-tree $\mathcal{T}$. To keep our notation simple, we will assume (by introducing new edges, if necessary) that (a) all processors in A are adjacent, and (b) every processor in A_j is adjacent to every processor in I_j. Let T be the total number of transmissions in Π_2.

Notation: Let $b_1, b_2, \ldots, b_T$ be the sequence of bits transmitted in Π_2 by the auxiliary processors. Suppose, b_i is transmitted by vertex $v \in I_j$ by computing $g_i(b_1 b_2 \cdots b_{i-1}, x_j \oplus z_v)$, where x_j is the the restriction of the input assignment to I_j and z_v is an ϵ-noisy vector in $\{0,1\}^n$.

The tree: The nodes of the GKS-tree $\mathcal{T}$ are 0-1 sequences of length at most T (the root is the node at 0th level and corresponds to the empty sequence). The children of the node $b \in \{0,1\}^{i-1}$ $(0 \le i-1 \le T-1)$ are the two vertices $b0$ and $b1$. Fix $i \in [T]$ and suppose vertex $v \in A_j$ makes the i-th transmission. The function that v computes to determine what to transmit, will be used to compute the successor of the nodes at the $i-1$-th level. To state this formally, suppose the function computed by v is $g_i(b, x_j \oplus z_v)$, where $b \in \{0,1\}^{i-1}$, x_j is the restriction of the input to I_j, and z_v is an ϵ-noisy bit string of length n. Then, the label of $b \in \{0,1\}^{i-1}$ (at level $i-1$ in $\mathcal{T}$) is (j, h), where $h(x_j, z_v) = b \cdot g_i(b, x_j \oplus z_v)$. (Since our definition requires the function to return a child of b, h returns an extension of b in $\{0,1\}^i$.)

Advantage: The set of leaves of $\mathcal{T}$, $L(\mathcal{T})$, is precisely $\{0,1\}^T$. Let $a : L(\mathcal{T}) \to \{+1, -1\}$ be defined by $a(b_1 b_2 \cdots b_T) = (-1)^{b_T}$. Then, it follows from our definitions that

$$\begin{aligned}\mathsf{adv}_{\oplus,\mu}(\mathcal{T}) &\ge \mathrm{E}[\oplus(x) a(\mathcal{T}(x))] \\ &= \mathrm{E}[\oplus(x)(-1)^{b_T}] \\ &= \mathsf{adv}_{\oplus,\mu}(\Pi_2).\end{aligned}$$

Proof of Lemma 3.3. Follows immediately from Lemmas 5.1, 5.2 and 5.3.

References

[1] U. Feige and J. Kilian. Finding or in noisy broadcast network. *Information Processing Letters*, 73(1-2):69–75, January 2000.

[2] R. G. Gallager. Finding parity in simple broadcast networks. *IEEE Transactions on Information Theory*, 34:176–180, 1988.

[3] N. Goyal, G. Kindler, and M. E. Saks. Lower bounds for the noisy broadcast problem. In *Proc. of the 46th Annual IEEE Symposium on Foundations of Computer Science; full version available at www.math.rutgers.edu/ saks/PUBS/nb-submitted.pdf*, pages 40–52, October 2005.

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

[5] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[6] Y. Kanoria and D. Manjunath. On distributed computation in noisy random planar networks. In *Proc. of the IEEE International Symposium on Information Theory*, France, June 2007.

[7] E. Kushilevitz and Y. Mansour. Computation in noisy radio networks. In *Proceedings of the 9th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 236–243, 1998.

[8] I. Newman. Computing in fault tolerance broadcast networks. In *proc. of the 19th IEEE Annual Conference on Computational Complexity*, pages 113–122, 2004.

[9] L. Ying, R. Srikant, and G. Dullerud. Distributed symmetric function computation in noisy wireless sensor networks with binary data. In *Proc. of the 4th International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless networks (WiOpt)*, pages 1–9, April 2006.