Distributed Function Computation in Networks: a Joint Delay-Energy Perspective

Nikhil Karamchandani  
Massimo Franceschetti  
California Institute for Telecommunications and Information Technology  
Department of Electrical and Computer Engineering  
University of California at San Diego  
La Jolla, CA 92093-0407  
nikhil@ucsd.edu  
massimo@ece.ucsd.edu

Abstract—This paper considers the following network computation problem: $n$ nodes are placed on a $\sqrt{n} \times \sqrt{n}$ grid, each node is connected to every other node within distance $r(n)$ from itself, and is given an arbitrary input bit. Nodes communicate with each other so that finally a designated sink node can compute a target function $f$ of the input bits. We focus on computing the identity function and the class of symmetric functions under two different communication models. We first consider a noiseless model where links are independent and noise-free, suitable for modeling wired networks. Next, we study a noisy broadcast model in which when a node transmits a bit, each of its neighbors receives a noisy copy of the bit. This is a simple model for wireless communications, originally proposed by El Gamal (1987). We use the protocol model for interference and nodes which do not share neighbors are allowed to transmit simultaneously. For every connection radius $r(n)$, we present lower bounds on the minimum number of transmissions and the minimum number of time slots required to compute $f$. We then describe efficient protocols which can match both these lower bounds up to a constant factor.

I. INTRODUCTION

Sensor networks consist of a large number of small nodes, each capable of sensing, processing, and communicating data. Such networks are typically required to sample a field of interest, do ‘in-network’ computations, and then communicate a relevant summary of the data to a designated node(s), most often a function of the raw sensor measurements. For example, in environmental monitoring a relevant function might be the average temperature in a region. Another example is an intrusion detection network, where a node switches its message from 0 to 1 if it detects an intrusion and a useful function is the maximum of all the node messages. For designing protocols to perform such in-network aggregation, two relevant measures of efficiency are the latency and the energy cost. In this paper, we measure the latency of a protocol by the number of time slots it takes and the energy cost by the total number of transmissions made in the network.

Network computation has been studied extensively in the literature, under a wide variety of models. Networks with noiseless communication links are suitable for modeling wired networks. Computation in such networks has been studied traditionally in the context of communication complexity [1]. Wireless networks, on the other hand, have three distinguishing features: noise, the inherent broadcast medium, and interference. Computation in large wireless sensor networks has been studied recently in [2]–[4] but the model in these works excludes noise. This is a significant limitation since protocols designed for the noiseless setting might fail when used in more realistic noisy networks because of errors in decoding messages. A simple model encompassing the key features of wireless communications was proposed in [5] in which when any node transmits a bit, each of its neighbors receives an independent noisy copy of the bit. We use the protocol model [6] for interference and nodes which do not share neighbors are allowed to transmit simultaneously. Under this communication model, [7]–[9] consider computation in a complete network where each node is connected to every other node. An alternative to the complete network is the random geometric network in which $n$ nodes are randomly deployed in a $\sqrt{n} \times \sqrt{n}$ square and each node can communicate with all other nodes in a range $r(n)$. Computation in such networks under the noisy broadcast communication model has been studied in [10]–[13]. All of these works focus on the scenario when the connection radius $r(n)$ is $\Theta(\sqrt{\log n})$, which is the threshold to obtain a connected random geometric network, see for example [14, Chapter 3].

A drawback of previous work is that they study computation in specific networks and do not provide intuition about computation in general networks or how network structure impacts the cost of computation. As a first step towards rectifying this shortcoming, we define a class of deterministic geometric networks. These are constructed as follows. Consider a $\sqrt{n} \times \sqrt{n}$ grid of $n$ nodes. As shown in Figure 1, each node in the grid is connected to every other node which is within distance $r(n)$ from it. This construction has many useful features. Firstly, it can be used to model a regular deployment of a sensor network with radios of limited range $r(n)$. When $r(n) \geq \sqrt{2n}$, all nodes are connected to each other and the network reduces to the complete network. Furthermore, above the critical connectivity radius for the random geometric network $r(n) = \Theta(\sqrt{\log n})$, the network has structural properties similar to its random geometric counterpart and hence all the results in this paper also hold in that scenario. Thus our model includes the two network structures studied in previous works as special cases. Further, by varying the connectivity radius $r(n)$ we can study a broad variety of networks with contrasting structural properties, ranging from the sparsely
connected grid network for $r(n) = 1$ to the densely connected complete network when $r(n) \geq \sqrt{2n}$. As we will see, studying computation in this class of networks will provide intuition about how network properties like the average node degree impact cost of computation and help identify principles which can be used to design efficient schemes for general network topologies.

The main objective of this paper is to find schemes for computing in deterministic geometric networks, which are efficient both in terms of the number of time slots and the number of transmissions required for any connection radius $r(n) \in [1, \sqrt{2n}]$. We consider both the noiseless as well as the noisy broadcast communication models. For each case, we propose schemes to compute the identity function (i.e. recover all source messages) and symmetric functions.$^1$ We also show the (order) optimality of these schemes with respect to both the number of time slots and the number of transmissions required. Previous work in this direction includes [7], [8], and [13] which studied the computation of the identity function, but only for the case when $r = \sqrt{2n}$ and $r = \Theta(\sqrt{\log n})$ respectively. Symmetric functions were considered in [10], [12]. However, all of these works were solely concerned with minimizing the number of transmissions required for computation. We generalize the results to any connection radius $r(n) \in [1, \sqrt{2n}]$ and propose new schemes which are also delay-optimal, in addition to being efficient in terms of the number of transmissions required. In fact, most previous work ignores the delay aspect of network computation. References [4], [11] consider the question of delay for computing the maximum function, but again only for the case when $r(n) = \Theta(\sqrt{\log n})$.

The rest of the paper is organized as follows. We describe the problem setup in detail and give some preliminaries in Section II. Networks with noiseless links are discussed in Section III. Noisy networks are analyzed in Section IV. Finally, we conclude in Section V.

II. PROBLEM SETUP

Consider a $\sqrt{n} \times \sqrt{n}$ grid of $n$ nodes, as shown in Figure 1. For each node $i$ in the grid, the set of neighbors $N(i)$ consists of all the nodes which are within distance$^2$ $r(n)$ of $i$. There exists an edge between $i$ and every neighbor $j \in N(i)$, denoted by $(i, j)$. Edges in the network correspond to communication links. We denote such a network by $\mathcal{N}(n, r)$. Each node $i$ is assigned an input bit $x_i \in \{0, 1\}$. Let $x$ denote the vector whose $i$th component is $x_i$. We often refer to $x$ as the input to the network. The nodes communicate with each other so that a designated sink node $v^*$ can compute a target function $f$ of the input bits,

$$ f : \{0, 1\}^n \to \mathcal{B} $$

where $\mathcal{B}$ denotes the co-domain of $f$.$^3$

$^1$The value of a symmetric function is invariant to permutations of the input messages, for example the maximum function. A complete definition is provided in the next section.

$^2$In the sequel, we will suppress the dependence of $r(n)$ on $n$ and denote it simply by $r$.

Let time be divided into slots of unit duration. We consider two different communication models:

- **Noiseless model**: Under this model, if a node $i$ transmits a bit on an edge $(i, j)$ in a time slot, then node $j$ receives the bit without any error in the same slot. Further, all the edges in the network can be used simultaneously since there is no interference.

- **Noisy broadcast model** [8], [12]: Under this model, if a node $i$ transmits a bit $b$ in time slot $t$, then each neighboring node in $N(i)$ receives a noisy copy of $b$ in the same slot. More precisely, neighbor $j \in N(i)$ receives $b \oplus \eta_{i,j,t}$ where $\oplus$ denotes modulo-2 sum. $\eta_{i,j,t}$ is a bernoulli random variable that takes value 1 with probability $\epsilon$ and 0 with probability $1 - \epsilon$. The noise bits $\eta_{i,j,t}$ are independent over $i, j$ and $t$. Such a network is called an $\epsilon$-noise network.

We use the “protocol model” [6] for interference; two nodes $i$ and $j$ can transmit in the same time slot only if they do not have any common neighbors, i.e., $N(i) \cap N(j) = \emptyset$. Thus any node can receive at most one bit in a time slot. Note that in the classical protocol model [6], communication is reliable if simultaneously transmitting nodes are well-separated. However in our case, even with no interference there is still a probability of error $\epsilon$ which models the inherent noise in the wireless communication medium.

The nodes in the network take turns to transmit messages to their neighbors. A protocol for computing a target function $f$ specifies the order in which nodes in the network transmit and the procedure for each node to decide what to transmit in its turn. At the end of the communication, the sink node $v^*$ computes an estimate $\hat{f}$ for the value of the target function $f$.

A protocol is defined by the total number of time slots $T$ of its execution, and for each slot $t \in \{1, 2, \ldots, T\}$, a collection of $S_t$ simultaneously transmitting$^4$ nodes $\{v^t_1, v^t_2, \ldots, v^t_{S_t}\}$ and corresponding encoding functions $\{\phi^t_1, \phi^t_2, \ldots, \phi^t_{S_t}\}$. In any time slot $t \in \{1, 2, \ldots, T\}$, node $v^t_j$ computes the function $\phi^t_j : \{0, 1\} \times \{0, 1\}^d_j \to \{0, 1\}$ of its input bit and the $\eta^t_j$ bits it received before time $t$ and then transmits this value. After the $T$ rounds of communication, the sink node $v^*$ computes$^4$ an estimate $\hat{f}$ for the value of the function $f$. Note that in this model, both the duration $T$ and the total number of transmissions $\sum_{t=1}^T S_t$ required by a protocol are the same for all inputs $x \in \{0, 1\}^n$.

This model enforces the protocols to have certain properties which are desirable in noisy scenarios: First, the protocols are oblivious in the sense that in any time slot, the node which transmits is decided ahead of time and does not depend on a particular execution of the protocol. Without this property, the noise in the network may lead to multiple nodes transmitting in the same time slot. Thus the list of simultaneously transmitting nodes might contain repetitions.

$^3$In the noiseless case, a node $i$ will be included separately for each distinct edge on which it transmits in a given slot. Thus the list of simultaneously transmitting nodes might contain repetitions.

$^4$We are considering the one-shot case here, i.e., the sink computes the function for only one instance of source messages.
characterizing the minimum delay of broadcast communication models, we are interested in characterizing the minimum delay for target function \( f \) if for any distribution \( \mu \) on the input \( x \), \( \Pr(\hat{f} \neq f(x)) \leq \delta \). For both the noiseless and noisy broadcast communication models, we are interested in characterizing the minimum delay \( T \) and the minimum number of transmissions required by a \( \delta \)-error protocol for computing a target function \( f \) in the network \( N(n, r) \), as a function of the number of nodes \( n \) and the connection radius \( r \).

In this work, we consider two specific classes of functions. First, we consider the identity function, which is interesting because it can be used to compute any other function and thus gives a baseline to compare with when considering other functions. As in [2], [3], [10], we also study the class of symmetric functions. A function \( f \) is symmetric if for any input \( x \in \{0,1\}^n \) and permutation \( \pi \) on \( \{1,2,\ldots,n\} \),

\[
 f(x_1,x_2,\ldots,x_n) = f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}).
\]

In other words, the value of the function only depends on the arithmetic sum of the input arguments, i.e., \( \sum_{i=1}^n x_i \). Many functions which are useful in the context of sensor networks are symmetric, for example the average, maximum, majority and parity. We briefly summarize our results before moving on to the proofs. Under each communication model and for any connection radius \( r \in [1, \sqrt{2n}] \), we first prove lower bounds on the delay and the number of transmissions required by any protocol for computing the identity function. We then describe a protocol which matches these bounds up to a constant factor. Next, we consider the class of symmetric functions. For a particular symmetric target function (parity function), we provide lower bounds on the delay and the number of transmissions required by any protocol for computing the function. We then present a protocol which can compute any symmetric function while matching the above bounds up to a constant factor. Our results are summarized in Tables I and II. Note that our results clearly illustrate the effect of the average node degree \( \Theta(r^2) \) on the cost of computation under both communication models. Further, by comparing the results for the identity function and symmetric functions, one can quantify the gains in performance can be achieved by using in-network aggregation for computation. Finally, we would like to remark that though we present the analysis for deterministic geometric networks, all our results for the case when \( r \geq \Omega(\sqrt{\log n}) \) also hold for random geometric networks with \( n \) nodes distributed randomly in a \( \sqrt{n} \times \sqrt{n} \) square.

### A. Preliminaries

In this section, we present some preliminary observations which will be useful later.

**Remark II.1.** For connection radius \( r < 1 \), each node in the network \( N(n, r) \) is isolated and hence computation is infeasible. On the other hand, for any \( r \geq \sqrt{2n} \), the network \( N(n, r) \) is fully connected and does not change with increasing \( r \). Thus the interesting regime is when the connection radius \( r \in [1, \sqrt{2n}] \).

**Remark II.2.** For any connection radius \( r \in [1, \sqrt{2n}] \), each node in the network \( N(n, r) \) has \( \Theta(r^2) \) neighbors.

**Lemma II.3.** Consider the line network in Figure 2, let each source \( v_i \) have an \( m \)-bit message \( x_i \) to start with. Transmissions can be pipelined so that the node \( v_i \) can collect all the source messages in at most \( ml \) time slots and using a total of at most \( ml^2 \) transmissions.

**Theorem II.4.** (Gallager’s Coding Theorem) [10, Page 3, Theorem 2], [15]: For any \( \gamma > 0 \) and any integer \( m \geq 1 \), there exists a code for sending an \( m \)-bit message over a binary symmetric channel using \( O(m) \) transmissions such that the message is received correctly with probability at least \( 1 - e^{-\gamma m} \).

Finally, throughout the paper \( \log \) will denote logarithm to the base two.

### III. Noiseless Deterministic Geometric Networks

We begin with the identity function. We have the following straightforward lower bound.

**Theorem III.1.** Let \( f \) be the identity function, \( \delta \in [0,1/2) \) and connection radius \( r \in [1, \sqrt{2n}] \). Any \( \delta \)-error protocol for computing \( f \) over \( N(n, r) \) requires at least \( \Omega(n/r^2) \) time slots and \( \Omega(n^{3/2}/r) \) transmissions.
TABLE I
RESULTS FOR NOISELESS DETERMINISTIC GEOMETRIC NETWORKS.

<table>
<thead>
<tr>
<th>Function</th>
<th>No. of time slots</th>
<th>No. of transmissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>$\Theta(\frac{n}{r^2})$</td>
<td>$\Theta(n^{1/2}/r)$</td>
</tr>
<tr>
<td>Symmetric</td>
<td>$\Theta(\frac{n}{r})$</td>
<td>$\Theta(n)$</td>
</tr>
</tbody>
</table>

TABLE II
RESULTS FOR NOISY BROADCAST DETERMINISTIC GEOMETRIC NETWORKS.

<table>
<thead>
<tr>
<th>Function</th>
<th>No. of time slots</th>
<th>No. of Transmissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>$\max{\Theta(n), \Theta(r^2 \log \log n)}$</td>
<td>$\max{\Theta(n^{1/2}/r), \Theta(n \log \log n)}$</td>
</tr>
<tr>
<td>Symmetric</td>
<td>$\max{\Theta(n/r), \Theta(r^2 \log \log n)}$</td>
<td>$\max{\Theta(n \log n/r^2), \Theta(n \log \log n)}$</td>
</tr>
</tbody>
</table>

Proof: For computing the identity function, the sink node $v^*$ should receive at least $n-1$ bits. Since $v^*$ has $O(r^2)$ neighbors and can receive at most one bit on each edge in a time slot, it will require at least $\Omega\left(\frac{n}{r^2}\right)$ time slots to compute the identity function.

It is easy to verify that there exists a collection of $\Omega\left(\frac{n}{r}\right)$ disjoint cuts such that each cut separates $\Omega(n)$ sources from $v^*$, see Figure 3 for an example. Thus to ensure that $v^*$ can compute the identity function, there should be at least $\Omega(n)$ transmissions across each cut. The lower bound on the total number of transmissions then follows.

We now present a simple protocol for computing the identity function which is order-optimal in both the delay and the number of transmissions.

Theorem III.2. Let $f$ be the identity function and connection radius $r \in [1, \sqrt{n}]$. There exists a zero-error protocol for computing $f$ over $N(n, r)$ which requires at most $O\left(\frac{n}{r^2}\right)$ time slots and $O\left(n^{3/2}/r\right)$ transmissions.

Proof: Let $c = r/\sqrt{2}$. Consider a division of the network $N(n, r)$ into cells of size $c \times c$, see Figure 4. Note that each node is connected to all nodes in its own cell as well as in any neighboring cell. The protocol works in three phases, see Figure 4. In the first phase, source messages are horizontally aggregated towards the left-most column of cells along parallel linear chains. In the second phase, the messages in the left-most cells are vertically aggregated towards the nodes in the cell containing the sink node $v^*$. In the final phase, all the messages are collected at the sink node.

The first phase has messages aggregating along $O\left(\sqrt{n}r\right)$ parallel linear chains each of length $O\left(\sqrt{n}/r\right)$. By pipelining the transmissions, from Lemma II.3 this phase requires $O\left(\sqrt{n}/r\right)$ time slots and a total of $O\left(\sqrt{n}r \times n/r^2\right)$ transmissions in the network. Since each node in the left-most column of cells has $O\left(\sqrt{n}/r\right)$ one-bit messages and there are $O(r^2)$ parallel chains each of length $O\left(\sqrt{n}/r\right)$, the second phase uses $O\left(r^2 \times \sqrt{n}/r \times n/r^2\right)$ transmissions and $O\left(\sqrt{n}/r \times \sqrt{n}/r\right)$ time slots. In the final phase, each of the $O(r^2)$ nodes in the cell with $v^*$ has $O\left(n/r^2\right)$ one-bit messages and hence it requires $O(n)$ transmissions and $O\left(n/r^2\right)$ slots to finish. Adding the costs, the protocol can compute the identity function with $O\left(n^{3/2}/r\right)$ transmissions.
and \( O\left(\frac{n}{r^2}\right) \) time slots.

Now we consider the computation of symmetric functions. We have the following straightforward lower bound:

**Theorem III.3.** Let \( \delta \in [0, 1/2] \) and connection radius \( r \in [1, \sqrt{2n}] \). There exists a symmetric target function \( f \) such that any \( \delta \)-error protocol for computing \( f \) over \( \mathcal{N}(n, r) \) requires at least \( \Omega\left(\frac{n}{r}\right) \) time slots and \( n - 1 \) transmissions.

**Proof:** Let \( f \) be the parity function. For computing the parity function, each non-sink node in the network should transmit at least once. Hence, at least \( n - 1 \) transmissions are required. Further, since the message of the farthest node requires at least \( \Omega\left(\frac{n}{r}\right) \) time slots to reach \( v^* \), we have the lower bound on the duration of any protocol.

Next, we present a matching upper bound.

**Theorem III.4.** Let \( f \) be any symmetric function and connection radius \( r \in [1, \sqrt{2n}] \). There exists a zero-error protocol for computing \( f \) over \( \mathcal{N}(n, r) \) which requires at most \( O\left(\frac{n}{r}\right) \) time slots and \( O(n) \) transmissions.

**Proof:** We present a protocol which can compute the arithmetic sum of all the input bits over \( \mathcal{N}(n, r) \). Note that this suffices to prove the result.

Again, consider a division of the noiseless network \( \mathcal{N}(n, r) \) into cells of size \( c \times c \) with \( c = r/\sqrt{8} \). For each cell, pick one node arbitrarily and call it the "cell-center". For the cell containing \( v^* \), choose \( v^* \) to be the cell center. The protocol works in two phases, see Figure 5.

**First phase:** All the nodes in a cell transmit their input bits to the cell-center. This phase requires just one time-slot and \( n \) transmissions and at the end of the phase, each cell-center knows the arithmetic sum of the input bits in its cell, which is an element of \( \{0, 1, \ldots, \Theta\left(\frac{r^2}{2}\right)\} \).

**Second phase:** In this phase, the values at the cell-centers are aggregated so that \( v^* \) can compute the arithmetic sum of all the input bits in the network. There are two cases, depending on the connection radius \( r \).

- \( r \leq \sqrt{\log n} \): Since each cell-center is connected to the other cell-centers in its neighboring cells, this phase can be mapped to computing the arithmetic sum over the noiseless network \( \mathcal{N}\left(\Theta\left(\frac{n}{r^2}\right), 1\right) \) where each node observes a message in \( \{0, 1, \ldots, \Theta\left(\frac{r^2}{2}\right)\} \). See Figure 5(a) for an illustration. We design a protocol to complete this phase using \( O\left(\frac{n}{r^2}\right) \) transmissions and \( O\left(\frac{n}{r}\right) \) time slots.

- \( r > \sqrt{\log n} \): The messages at cell-centers are aggregated towards \( v^* \) along a tree, see Figure 5(b). The value at each cell-center can be viewed as a \( \lceil \log n \rceil \)-length binary vector. To transmit its vector to the parent (cell-center) node in the tree, every leaf node (in parallel) transmits each bit of the vector to a distinct node in the parent cell. In the next time slot, each of these intermediate nodes relays its received bit to the corresponding cell-center. The parent cell-center can then reconstruct the message and aggregate it with its own value to form another \( \lceil \log n \rceil \)-length binary vector. Note that it requires two time slots and \( O(\log n) \) transmissions by a cell-center to traverse one level of depth in the aggregation tree. This step is performed repeatedly (in succession) till the sink node \( v^* \) receives the sum of all the input bits in the network. Since the depth of the aggregation tree is \( \Omega\left(\frac{n}{r}\right) \), the phase requires \( O\left(\frac{n}{r}\right) \) time slots. There are \( O(\log n) \) transmissions in each cell of the network. Hence the phase requires a total of \( O\left(\frac{n}{r^2} \log n\right) \) transmissions.

Adding the costs of the two phases, the protocol can compute any symmetric function using \( O(n) \) transmissions and \( O\left(\frac{n}{r}\right) \) time slots.

**IV. Noisy Broadcast Deterministic Geometric Networks**

Recall that each edge in an \( \epsilon \)-noise network is a binary symmetric channel with parameter \( \epsilon \in (0, 1/2) \). Also recall that we use the protocol model for managing interference due to simultaneous transmissions in a noisy broadcast network.

To begin with, we consider the computation of the identity function. We have the following lower bound.

**Theorem IV.1.** Let \( f \) be the identity function. Let \( \delta \in (0, 1/2) \), \( \epsilon \in (0, 1/2) \) and connection radius \( r \in [1, \sqrt{2n}] \). Any \( \delta \)-error protocol for computing \( f \) over an \( \epsilon \)-noise network \( \mathcal{N}(n, r) \) requires at least \( \max\{n - 1, \Omega\left(\frac{r^2}{2} \log \log n\right)\} \) time slots and \( \max\{\Omega\left(\frac{n^{3/2}}{r}\right), \Omega(n \log \log n)\} \) transmissions.

**Proof:** The lower bound of \( \Omega\left(\frac{n^{3/2}}{r}\right) \) transmissions follows from the same argument as in the proof of Theorem III.1. The other lower bound of \( \Omega(n \log \log n) \) transmissions follows from [8, Corollary 2].

We now turn to the number of time slots required. For computing the identity function, the sink node \( v^* \) should receive at least \( n - 1 \) bits. However, the sink can receive at most one bit in any slot and hence any protocol for computing the identity function requires at least \( n - 1 \) time slots. For the remaining lower bound, consider a division of the network \( \mathcal{N}(n, r) \) into cells of size \( c \times c \) with \( c = r/\sqrt{8} \). Since the total number of transmissions in the network is at least \( \Omega(n \log \log n) \) and there are \( O\left(\frac{n}{r^2}\right) \) cells, there is at least one cell where the number of transmissions is at least \( \Omega\left(\frac{r^2}{2} \log n\right) \). Since all nodes in a cell are connected to each other, at most one of them can transmit in a slot. Thus any protocol for computing the identity function requires at least \( \Omega\left(\frac{r^2}{2} \log n\right) \) time slots.

Next, we present an efficient protocol for computing the identity function in noisy broadcast networks, which matches the above bounds.

**Theorem IV.2.** Let \( f \) be the identity function. Let \( \delta \in (0, 1/2) \), \( \epsilon \in (0, 1/2) \) and connection radius \( r \in [1, \sqrt{2n}] \). There exists a \( \delta \)-error protocol for computing \( f \) over an \( \epsilon \)-noise network \( \mathcal{N}(n, r) \) which requires at most \( \max\{O(n), O\left(\frac{r^2}{2} \log \log n\right)\} \) time slots and \( \max\{O\left(\frac{n^{3/2}}{r}\right), O(n \log \log n)\} \) transmissions.

**Proof:** Consider a division of the network \( \mathcal{N}(n, r) \) into cells of size \( c \times c \) with \( c = r/\sqrt{8} \). As discussed before, to
limit interference in a noisy network two nodes are allowed to transmit in the same time slot only if they do not have any common neighbors. Cells are scheduled according to the scheme shown in Figure 6 to ensure that all transmissions are successful. Thus, each cell is scheduled once every $7 \times 7$ time slots. Within a cell, at most one node can transmit in any given time slot and nodes take turns to transmit one after the other. For each cell, pick one node arbitrarily and call it the “cell-center”. The protocol works in three phases, see Figure 7.

**First phase:** There are two different cases, depending on the connection radius $r$.

- $r \leq \sqrt{n}/\log n$: In this case, each node in its turn transmits its input bit to the corresponding cell-center using a codeword of length $O(\log n)$ such that the cell-center decodes the message correctly with probability at least $1 - 1/n^2$. The existence of such a code is guaranteed by Theorem II.4. This phase requires at most $O(r^2 \log n)$ time slots and at most $O(n \log n)$ transmissions in the network. Since there are $O(n/r^2)$ cells in the network, the phase executes correctly in all cells of the network with probability of error at most $O(1/n)$.
- $r \geq \sqrt{n}/\log n$: In this case, each cell uses the more sophisticated protocol described in [8, Section 7] for recovering all the input messages from the cell at the cell-center. This protocol requires at most $O(r^2 \log \log n)$ time slots and
a total of at most \( O \left( n/r^2 \times r^2 \log \log n \right) \) transmissions in the network. At the end of the protocol, a cell-center has all the input messages from its cell with probability at least \( 1 - \log n/n \). Since there are at most \( \log^2 n \) cells in the network for this case, the phase executes correctly in all cells of the network with probability of error at most \( \log^3 n/n \).

Thus at the end of the first phase, each cell-center in the network has all the input messages of the nodes in its cell with high probability.

**Second phase:** In this phase, the messages collected at the cell-centers are aggregated horizontally towards the left-most cells, see Figure 7. Note that there are \( \sqrt{n}/r \) horizontal chains and each cell-center has \( O \left( r^2 \right) \) input messages. In each such chain, the rightmost cell-center maps its set of messages into a codeword of length \( O \left( \sqrt{n}r \right) \) and transmits it to the next cell-center in the horizontal chain. The receiving cell-center decodes the incoming codeword, re-encodes it into a codeword of length \( O \left( \sqrt{n}r \right) \), and then transmits it to the next cell-center, and so on. This phase requires at most \( O \left( \sqrt{n}r \times \sqrt{n}/r \right) \) time slots and a total of at most \( O \left( \sqrt{n}r \times n/r^2 \right) \) transmissions in the network. From Theorem II.4, this step can be executed throughout the network with probability of error at most \( O \left( 1/n \right) \).

**Third phase:** In the final phase, the messages collected at the cell-centers of the left-most column are aggregated vertically towards the sink node \( v^* \), see Figure 7. Each cell-center maps its set of input messages into a codeword of length \( O \left( \sqrt{n}r \right) \) and transmits it to the next cell-center in the chain. The receiving cell-center decodes the incoming message, re-encodes it, and then transmits it to the next node, and so on. By pipelining the transmissions, from Lemma II.3 this phase requires at most \( O \left( \sqrt{n}r \times \sqrt{n}/r \right) \) time slots and at most \( O \left( \sqrt{n}r \times n/r^2 \right) \) transmissions in the network. This phase can also be executed with probability of error at most \( O \left( 1/n \right) \).

At the end of the three phases, the sink node \( v^* \) can compute the identity function with high probability. Adding the costs of the phases, the protocol requires at most \( \max\{O \left( n \right), O \left( r^2 \log \log n \right) \} \) time slots and \( \max\{O \left( n^{3/2}/r \right), O \left( n \log \log n \right) \} \) transmissions.

We now discuss the computation of symmetric functions in noisy broadcast networks. We begin with a lower bound on the delay and the number of transmissions required.

**Theorem IV.3.** Let \( \delta \in (0, 1/2) \), \( \epsilon \in (0, 1/2) \) and connection radius \( r \in [1, n^{1/2-\beta}] \) for any \( \beta > 0 \). There exists a symmetric target function \( f \) such that any \( \delta \)-error protocol for computing \( f \) over an \( n \)-noise network \( \mathcal{N}(n, r) \) requires at least \( \max\{\Omega \left( \sqrt{n}/r \right), \Omega \left( r^2 \log \log n \right) \} \) time slots and \( \max\{\Omega \left( n \log \log n \right), \Omega \left( n \log \log n \right) \} \) transmissions.

**Proof:** Let \( f \) be the parity function. The lower bound on the number of transmissions follows directly from [17, Theorem IV.2]. Since the message of the farthest node requires at least \( \Omega \left( \sqrt{n}/r \right) \) time slots to reach \( v^* \), we have the corresponding lower bound on the duration of any \( \delta \)-error protocol. The lower bound of \( \Omega \left( r^2 \log \log n \right) \) time slots follows from the same argument as in the proof of Theorem IV.1.

We now present an efficient protocol for computing any symmetric function in a noisy broadcast network which matches the above lower bounds.

**Theorem IV.4.** Let \( f \) be any symmetric function. Let \( \delta \in (0, 1/2) \), \( \epsilon \in (0, 1/2) \) and connection radius \( r \in [1, \sqrt{2n}] \). There exists a \( \delta \)-error protocol for computing \( f \) over an \( r \)-noise network \( \mathcal{N}(n, r) \) which requires at most \( \max\{O \left( \sqrt{n}/r \right), O \left( r^2 \log \log n \right) \} \) time slots and \( \max\{O \left( n \log \log n \right), O \left( n \log \log n \right) \} \) transmissions.

**Proof:** We present a protocol which can compute the arithmetic sum of the input bits over \( \mathcal{N}(n, r) \). Note that this suffices to prove the result.

Consider a division of the network \( \mathcal{N}(n, r) \) into cells of size \( c \times c \) with \( c = r/\sqrt{8} \). For each cell, we pick one node arbitrarily and call it the “cell-center”. As discussed before, cells are scheduled to prevent interference between simultaneous transmissions, see Figure 6. The protocol works in three phases.

**First phase:** The objective of the first phase is to ensure that each cell-center computes the arithmetic sum of the input messages from the corresponding cell. Depending on the connection radius \( r \), this is achieved differently.

- \( r \leq \sqrt{\log n / \log \log n} \): We design a scheme which can compute the partial sums at each cell-center with high probability using \( O \left( n/r^2 \times \log n \right) \) total transmissions and \( O \left( \log n \right) \) time slots. Details of the scheme can be found in [16].
- \( r > \sqrt{\log n / \log \log n} \): In this case, we first divide each cell further into smaller sub-cells with \( \Theta \left( \log n / \log n \right) \) nodes each, see Figure 8. Each sub-cell has an arbitrarily chosen “head” node. In each sub-cell, we use the Intra-cell protocol from [10, Section III] to compute the sum of the input bits from the sub-cell to the corresponding head node with high probability. This requires \( O \left( \log \log n \right) \) transmissions from
each node in the sub-cell. Since there are $O(r^2)$ nodes in each cell and only one node in a cell can transmit in a time slot, this step requires $O(r^2 \log \log n)$ time slots and a total of $O(n \log \log n)$ transmissions in the network.

Next, each head node encodes the sum of the input messages from its sub-cell into a codeword of length $O(\log n)$ and transmits it to the corresponding cell-center. This requires a total of $O(n \log \log n)$ transmissions in the network and $O(r^2 \log \log n)$ time slots.

The received values are aggregated so that at the end of the first phase, each cell-cell center knows the sum of the input messages in its cell with high probability. For this case, the first phase requires $O(n \log \log n)$ transmissions in the network and $O(r^2 \log \log n)$ time slots to complete.

Second phase: In this phase, the partial sums stored at the cell-centers are aggregated along a tree (see for example, Figure 7) so that the sink node $v^*$ can compute the sum of all the input bits in the network. We have the following two cases, depending on the connection radius $r$.

- $r \geq (\sqrt{n} \log n)^{1/3}$: For this regime, our aggregation scheme is similar to the Inter-cell protocol in [10, Section III]. Each cell-center encodes its message into a codeword of length $\Theta(\log n)$. Each leaf node in the aggregation tree sends its codeword to the parent node which decodes the message, sums it with its own message and then re-encodes it into a codeword of length $\Theta(\log n)$. The process continues till the sink node $v^*$ receives the sum of all the input bits in the network. In this case, the phase requires $O(n \log n/r^2)$ transmissions in the network and $O(\sqrt{n}/r \log n)$ time slots.

- $r \leq (\sqrt{n} \log n)^{1/3}$: In this regime, the above simple aggregation scheme does not match the lower bound for delay in Theorem IV.3. A more sophisticated aggregation scheme is presented in [11, Section V], which uses ideas from [18] to efficiently simulate a scheme for noiseless networks in noisy networks. This scheme requires $O(n \log n/r^2)$ transmissions in the network and $O(\sqrt{n}/r)$ time slots.

Thus using the above protocol, the sink node $v^*$ can compute any symmetric function with high probability. Combining the costs of the two phases, it requires at most $\max\{O(\sqrt{n}/r), O(r^2 \log \log n)\}$ time slots and $\max\{O(n \log n/r^2), O(n \log \log n)\}$ transmissions.

V. CONCLUSIONS

Under both the noiseless and the noisy broadcast communication models, and for any connection radius $\rho(n)$, we have characterized up to order the minimum number of transmissions and the minimum amount of time required for computation in a deterministic geometric network when the target function is the identity function or is a symmetric function. Our results for the case when $r = \Omega(\log n)$ also hold for random geometric networks. In addition, they provide intuition about how network properties like the average node degree impact cost of computation and help identify principles which can be used to design efficient schemes for general network topologies. In our setup here, we assumed that each node in the network observes either a 0 or a 1. Our results can be readily adapted to get upper bounds on the delay and number of transmissions required for the more general scenario where each node $i$ observes a block of input messages $x_{i1}^1, x_{i2}^2, \ldots, x_i^q$ with each $x_i^j \in \{0, 1, \ldots, q\}$, $q \geq 2$. However, finding matching lower bounds is more challenging and will be the subject of future work. Alternate directions for further research include the study of other interesting, possibly non-symmetric, target functions and more general network models.

REFERENCES


