Energy-Latency Tradeoff for In-Network Function Computation in Random Networks

Paul Balister*, Béla Bollobás*, Animashree Anandkumar†, Alan Willsky‡

January 12, 2011

Abstract

The problem of designing policies for in-network function computation with minimum energy consumption subject to a latency constraint is considered. The scaling behavior of the energy consumption under the latency constraint is analyzed for random networks, where the nodes are uniformly placed in growing regions and the number of nodes goes to infinity. The special case of sum function computation and its delivery to a designated root node is considered first. A policy which achieves order-optimal average energy consumption in random networks subject to the given latency constraint is proposed. The scaling behavior of the optimal energy consumption depends on the path-loss exponent of wireless transmissions and the dimension of the Euclidean region where the nodes are placed. The policy is then extended to computation of a general class of functions which decompose according to maximal cliques of a proximity graph such as the k-nearest neighbor graph or the geometric random graph. The modified policy achieves order-optimal energy consumption albeit for a limited range of latency constraints.

Keywords: Function computation, latency-energy tradeoff, Euclidean random graphs, minimum broadcast problem.

1 Introduction

A host of emerging networks are pushing the boundaries of scale and complexity. Data centers are being designed to distribute computation over thousands of machines. Sensor networks are being deployed in larger sizes for a variety of environmental monitoring tasks. These emerging networks face numerous challenges and the threat of a "data deluge" is an important one. The data collected by these networks typically scale rapidly as their size grows. Routing all the raw data generated in these large networks is thus not feasible and has poor scaling of resource requirements.

In this paper, we consider the scenario where only a function of the collected raw data is required at some specific node in the network. Many network applications fall into this category. For instance,

*P. Balister and B. Bollobás are with the Dept. of Math., Univ. of Memphis, Memphis, TN, USA Email: {pbalistr@,bollobas@msci.}memphis.edu. They are supported in part by NSF grants DMS-0906634, CNS-0721983 and CCF-0728928, and ARO grant W911NF-06-1-0076.

†A. Anandkumar is with the Center for Pervasive Communications and Computing, Electrical Engineering and Computer Science Dept., University of California, Irvine, USA 92697. Email: a.anandkumar@uci.edu. She is supported by the setup funds at UCI.

‡A.S. Willsky is with the Dept. of EECS, Massachusetts Institute of Technology, Cambridge, MA, USA. Email: willsky@mit.edu. He is supported in part by a MURI funded through ARO Grant W911NF-06-1-0076.

in a *statistical inference* application, where a decision has to be made based on the collected data, the *likelihood* function suffices to make the optimal decision [1]. Such functions can have significantly lower dimensions than the raw data and can thus considerably reduce the resource requirements for routing.

In this paper, we analyze the scaling behavior of energy and latency for routing and computation of functions in random networks, where the nodes are placed uniformly in growing regions and the number of nodes goes to infinity. In particular, we address the following questions: how can we exploit the structure of the function to reduce energy consumption and latency? What class of functions can be computed efficiently with favorable scaling of energy and latency requirements? How do the network properties such as the signal propagation model affect the scaling behavior? What is the complexity for finding the optimal policy with minimum energy consumption under a given latency constraint for function computation? Are there simple and efficient policies which achieve order optimal energy consumption? The answers to these questions provide important insights towards engineering in-network computation in large networks.

1.1 Summary of Contributions

The contributions of this paper are three-fold. First, we propose policies with efficient energy consumption which compute any function belonging to a certain structured class subject to a feasible latency constraint. Second, we prove order-optimality of the proposed policies in random networks. Third, we derive scaling laws for energy consumption in different regimes of latency constraints for different network models. To the best of our knowledge, this is the first work to analyze energy-latency tradeoff for function computation in large networks. These results provide insight into the nature of functions which are favorable for in-network computation.

We analyze the scaling laws for energy and latency in random networks, where n nodes are placed uniformly in a region of volume (or area) n in $\mathbb{R}^d$, and we let $n \to \infty$. We consider (single-shot) function computation and its delivery to a designated root node. We first consider the class of sum functions, which can be computed via an aggregation tree. We characterize the structural properties of the minimum latency tree and propose an algorithm to build an energy-efficient minimum latency tree based on successive bisection of the region of node placement. However, minimum latency comes at the expense of energy consumption and we relax the minimum latency constraint. Our modified algorithm achieves order-optimal energy consumption for any given latency constraint. It is based on the intuition that long-range communication links lower latency but increase energy consumption and the key is to strike a balance between having long-range and short-range communications to achieve the optimal tradeoff.

We then consider the more general class of functions that decompose as a sum of functions over the maximal cliques of a proximity graph, such as the k-nearest neighbor graph or the random geometric graph. These functions are relevant in the context of statistical inference of correlated measurements which are drawn from a *Markov random field*. See [1] for details. We extend the proposed sum-function policy to this case and prove that it achieves order-optimal energy consumption (up to logarithmic factors) albeit under a limited range of latency constraints. In this range of feasible latency constraints, the energy consumption is of the same order as sum function computation. Hence, functions based on locally-defined proximity graphs can be computed efficiently with optimal scaling of energy and latency requirements.

We analyze the scaling behavior of energy consumption under different regimes of latency constraints and for different signal propagation models. We assume that the energy consumed scales as R^ν where R is the inter-node distance and ν is the path-loss exponent and consider nodes placed in a region in $\mathbb{R}^d$. We prove that in the regime $1 \le \nu < d$, order-optimal energy consumption and minimum latency can both be achieved simultaneously. On the other hand, in the regime $\nu > d$, there is a tradeoff between energy consumption and the resulting latency of computation, and our policy achieves order-optimal tradeoff.

1.2 Prior and Related Work

There is extensive literature on in-network processing. Some of the earliest arguments for in-network processing for scalability are presented in [2, 3]. The work of Giridhar and Kumar [4] provides a theoretical framework for in-network computation of certain functions such as sum function and analyze scaling of capacity as the network size grows. However, the work in [4] is concerned with the rate of information flow when the function is computed an infinite number of times, while we consider latency of single-shot function computation. Single-shot computation is relevant in applications involving one-time decision making based on a set of measurements. Moreover, we consider a richer class of functions which decompose according to some proximity graph. These are relevant in statistical inference applications with correlated measurements.

For the special case of sum-function computation, the minimum latency is the same as that for the minimum broadcast problem, where the root has information that needs to be disseminated to all the nodes. Most of the previous work on minimum broadcast problem, e.g., [5, 6], have focused on obtaining good approximations for minimum latency in arbitrary networks, but do not address the issue of scaling behavior of latency-energy tradeoff in random networks. These works also assume that only short-range communication may be feasible for communication. On the other hand, we allow for a few long-range links but focus on obtaining favorable scaling of overall energy consumption. Works considering latency-energy tradeoff in multihop networks are fewer. For instance, the works in [7–9] consider energy-latency tradeoff for data collection but without the possibility of in-network computation, which can be significantly more expensive. The work in [10] considers latency-energy tradeoff but during the deployment phase of the network.

With respect to analysis of energy scaling laws in randomly placed networks, the work in [11] derives scaling laws for multihop routing without in-network computation. In [12], the minimum energy policy for graph-based function computation is first analyzed in the context of statistical inference of correlated measurements and is shown to be NP-hard. An efficient policy is derived based on the Steiner-tree approximation. In [1], scaling laws for energy consumption are derived for computation of graph-based functions in random networks. When the function decomposes according to the cliques of a proximity graph, such as the k-nearest neighbor graph or the random geometric graph, it is shown that the function can be computed with[1] $\Theta(n)$ energy consumption in random networks, where n is the number of nodes. A simple two-stage computation policy achieves this scaling and is shown to have asymptotically a constant approximation ratio, compared to the minimum energy policy. In this paper, we extend the work to incorporate latency constraints and design policies which minimize energy consumption under the constraints.

[1]For any two functions $f(n), g(n)$, $f(n) = O(g(n))$ if there exists a constant c such that $f(n) \le cg(n)$ for all $n \ge n_0$ for a fixed $n_0 \in \mathbb{N}$. Similarly, $f(n) = \Omega(g(n))$ if there exists a constant c' such that $f(n) \ge c'g(n)$ for all $n \ge n_0$ for a fixed $n_0 \in \mathbb{N}$, and $f(n) = \Theta(g(n))$ if $f(n) = \Omega(g(n))$ and $f(n) = O(g(n))$.

2 System Model

2.1 Communication and Propagation Model

In a wireless sensor network, there are communication and energy constraints. We assume that any node cannot transmit and receive at the same time (half duplex nodes). We assume that a node cannot receive from more than one transmitter at the same time and similarly, a node cannot transmit simultaneously to more than one receiver. We assume that no other interference constraints are present. This is valid if nearby nodes transmit in orthogonal channels or when they have idealized directional antenna which can focus the transmissions within a narrow region around the receiver (e.g., [13, 14]). We also assume that nodes are capable of adjusting their transmission power depending on the location of the receiver leading to better energy efficiency.

We assume unit propagation delays along all the communication links and negligible processing delays due to in-network computation at nodes. For a transmission along edge (i, j) (from node i to node j), the energy consumption[2] is equal to $R^{\nu}_{i,j}$, where $R_{i,j}$ is the Euclidean distance and typically $\nu \in [2, 6]$ for wireless transmissions. In this paper, we allow for any $\nu \geq 1$.

2.2 Stochastic model of sensor locations

Let $Q_n \subset \mathbb{R}^d$ denote the d-dimensional hypercube $[0, n^{1/d}]^d$ of volume n, and typically $d = 2$ or 3 for sensors placed in an Euclidean region. We assume that n sensor nodes (including the root) are placed uniformly in Q_n with sensor i located at $V_i \in \mathbb{R}^d$. We denote the set of locations of the n sensors by $\mathbf{V}_n{:=}\{V_1, \ldots, V_n\}$. For our scaling law analysis, we let the number of sensors $n \to \infty$. Denote the root node by r, where the computed function needs to be delivered, and its location by V_r.

2.3 Function Computation Model

Each sensor node i collects a measurement $Y_i \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite set, and let $\mathbf{Y}_n = \{Y_1, \ldots, Y_n\}$ be the set of measurements of n nodes. We assume that the goal of data aggregation is to ensure that a certain deterministic function[3] $\Psi : (\mathbf{Y}_n, \mathbf{V}_n) \mapsto \mathbb{R}$ is computable at the root r at the end of the aggregation process. The set of valid aggregation policies π is thus given by

$$\mathfrak{F}(\mathbf{V}_n; \Psi){:=}\{\pi : \Psi(\mathbf{Y}_n, \mathbf{V}_n) \text{ computable at } r\}. \tag{1}$$

Using the propagation model discussed in Section 2.1, the total energy consumption of the aggregation process under a policy $\pi \in \mathfrak{F}(\mathbf{V}_n; \Psi)$ is

$$\mathcal{E}^{\pi}(\mathbf{V}_n) := \sum_{e \in G^{\pi}_n} R^{\nu}_e, \tag{2}$$

where G^{π}_n is the set of links used for inter-node communication by the policy. The latency[4] of function computation is

$$L^{\pi}(\mathbf{V}_n; \Psi){:=} \inf[t : \Psi(\mathbf{Y}_n, \mathbf{V}_n) \text{ computable at } V_1 \text{ at time } t], \tag{3}$$

[2]Since nodes only communicate a finite number of bits, we use energy instead of power as the cost measure.
[3]In general, the function can depend on the locations where the measurements are collected.
[4]We consider one-shot function computation.

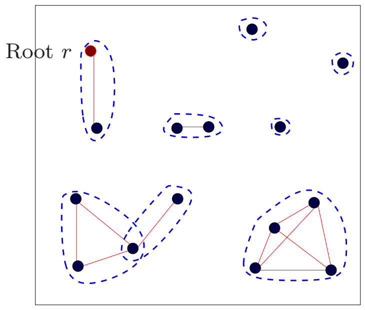

Figure 1: Example of a function dependency graph $\mathcal{G}$ and the function decomposes in terms of the maximal cliques of the graph, as represented by dotted lines.

where the aggregation process starts at $t = 0$. Let $L^*(\mathbf{V}_n; \Psi)$ be the minimum latency over the set of valid policies.

If no further assumptions are made on the function Ψ, then all the measurements $\mathbf{Y}_n$ need to be delivered to the root without any in-network computation. This is expensive both in terms of latency and energy consumption. Typically, the function Ψ decomposes into sub-functions involving only subset of measurements. In this case, in-network computation can be carried out to enable efficient tradeoff between energy consumption and latency of computation. We assume that the function Ψ has the form,

$$\Psi(\mathbf{V}_n, \mathbf{Y}_n) = \sum_{c \in \mathcal{C}} \psi_c((Y_i)_{i \in c}), \tag{4}$$

where $\mathcal{C}$ is the set of *maximal cliques*[5] on some graph $\mathcal{G}_\Psi$. See Fig.1 for an example. Note that this graph $\mathcal{G}_\Psi$ is related to the function Ψ and not with communication links. We refer to $\mathcal{G}_\Psi$ as the *function dependency graph*.

We consider the case when the graph[6] $\mathcal{G}$ is either a k-nearest neighbor graph (k-NNG) or the ρ-random geometric graph (ρ-RGG) with threshold radius ρ, where k, ρ are some fixed constants, independent of the number of nodes n. These graphs are relevant choices since many functions are based on proximity of the nodes. For instance, in the context of statistical inference, this corresponds to node measurements being locally dependent according to a *Markov random field* with the given graph $\mathcal{G}(\mathbf{V}_n)$. See [1] for details.

2.4 Energy-Latency Tradeoff

Denote the minimum latency for function computation over the set of valid policies by L^*, i.e.,

$$L^*(\mathbf{V}_n; \mathcal{G}_\Psi) := \min_{\pi \in \mathfrak{F}} L^\pi(\mathbf{V}_n; \mathcal{G}_\Psi). \tag{5}$$

The policy achieving minimum latency L^* can have large energy consumption and similarly, policies with low energy consumption can result in large latency. Hence, it is desirable to have policies that

[5]A clique is a complete subgraph and is maximal if it is not contained in a bigger clique.

[6]In fact, our results hold for a general class of graphs satisfying a certain stabilization property. See [15] for details and examples.

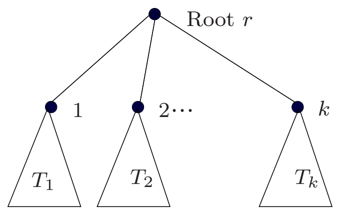

Figure 2: The latency for an aggregation tree can be obtained iteratively by considering subtrees. See Proposition 1.

can tradeoff between energy consumption and the latency of function computation. We consider finding a policy with minimum energy consumption subject to a latency constraint,

$$\mathcal{E}^*(\mathbf{V}_n;\delta,\mathcal{G}_\Psi):=\min_{\pi\in\mathfrak{F}}\mathcal{E}^\pi(\mathbf{V}_n;\mathcal{G}_\Psi),\quad s.t.\ L^\pi\leq L^*+\delta, \tag{6}$$

where δ (which can be a function of n) is the additional latency suffered in order to reduce energy consumption. In general, finding (6) is NP-hard for nodes placed at arbitrary locations (since the special case of this problem of finding minimum energy policy with no latency constraints is NP-hard [16]). We instead propose a policy which has energy consumption of the same order as the optimal policy for randomly placed nodes $\mathbf{V}_n$, as $n\to\infty$, and for any given latency constraint.

3 Sum Function Computation

A sub-class of functions in (15) is the set of sum functions

$$\Psi(\mathbf{V}_n,\mathbf{Y}_n)=\sum_{i=1}^n\psi_i(Y_i), \tag{7}$$

which have the maximum extent of decomposition over the set of nodes. Computing sum functions is required in various network applications, e.g., to find the average value, in distributed statistical inference with statistically independent measurements [1], and so on.

3.1 Preliminaries

We first discuss the policy to achieve minimum latency $L^*(\mathbf{V}_n;\Psi)$ in (5) for sum function computation without considering the energy consumption. In this case, the minimum latency does not depend on the position of the nodes $\mathbf{V}_n$ but only on the order of scheduling of the various nodes, i.e., $L^*(\mathbf{V}_n;\Psi)=L^*(n)$. Moreover, the minimum latency $L^*(n)$ can be achieved via data aggregation along a spanning tree $T^*(n)$, directed towards root r.

For data aggregation along any directed spanning tree T, each node waits to receive data from its children (via incoming links), computes the sum of the values (along with its own measurement) and then forwards the resulting value along the outgoing link. See Fig.3 for an example. Let L_T be the resulting latency along tree T. We now make a simple observation. See also Fig.2.

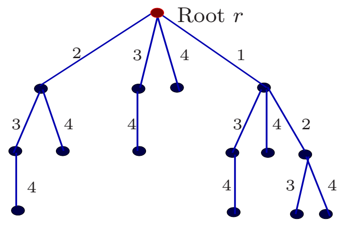

Figure 3: The min. latency tree T^* with edge level labels. See Alg.1.

Proposition 1 (Latency along a tree) *For a spanning tree T with root r, the latency L_T is given by*

$$L_T = \max_{i=1,\ldots,k} \{i + L_{T_i}\}, \tag{8}$$

where T_i is the subtree rooted at node i, and $1, \ldots, k$ are the children $\mathcal{C}(r;T)$ of the root node r ordered such that $L_{T_1} \geq L_{T_2} \ldots \geq L_{T_k}$.

Proof: Indeed, after time L_{T_i}, information from $1, \ldots, i$ has still not been sent to the root, and this will take at least time i, so $L_T \geq i + L_{T_i}$ for all $i = 1, \ldots, k$. Conversely, there is a simple policy with latency in (8) which aggregates along the subtrees T_i with latency L_{T_i} and then node i sends its data to the root r at time slot $L_T - i$. □

Using (8) we can thus effectively compute the latency of any given rooted tree T. We now provide the result on the minimum latency $L^*(n)$ and the construction of the tree $T^*(n)$ achieving it. This has been previously analyzed in the context of minimum broadcast problem [5], where the root has information that needs to be disseminated to all the nodes.

Lemma 1 (Minimum Latency Tree) *The minimum latency for sum function computation over n nodes is $L^*(n) = \lceil \log_2 n \rceil$. Equivalently, the maximum number of vertices in a tree with latency at most L is 2^L.*

Proof: See Appendix A. □

There is a unique minimum latency tree[7] $T^*(n)$ up to a permutation on the nodes. The minimum latency tree can be constructed recursively as explained in Algorithm 1. The algorithm runs for $L^*(n)$ steps and in each step, a child is added to each node already in the tree. An example of the minimum latency tree is shown in Fig.3.

3.2 Policies for Energy Latency Tradeoff

We now propose a policy for sum function computation with order-optimal energy consumption subject to a given latency constraint in (6). Note that the minimum latency tree $T^*(n)$ does not depend on the node locations and any permutation of the nodes on the tree (with the root fixed) results in the same latency. On the other hand, the energy consumption depends on the node locations. We propose an energy-efficient minimum-latency tree $T^*(\mathbf{V}_n)$ in Algorithm 2, depending on the node locations. This will be later proven to achieve order-optimal energy consumption for uniformly placed nodes. We first note some definitions used in the sequel.

[7]Note that the balanced binary tree on n nodes has latency $2\lceil \log_2(n+1) \rceil - 2$, which is about twice $L^*(n)$.

Algorithm 1 Min. latency tree $T^*(n)$.

Input: nodes $N = \{1, \ldots, n\}$, root node r. $\mathcal{C}(i;T)$ denotes children of node i. $\mathcal{S}(k;T)$ denotes level k edges in T. For any set A, let $A \overset{\cup}{\leftarrow} \{r\}$ denote $A \leftarrow A \cup \{r\}$.

Output: $T^*(n)$.

```
1: Initialize set $A = \{r\}$ and $T^* = \{r\}$.
2: for $k = 1, \ldots, \lceil \log_2 n \rceil$ do
3:   $B \leftarrow A$.
4:   for each $i \in B$ do
5:     if $N \setminus A \neq \emptyset$ then
6:       For some $j \in N \setminus A$, $\mathcal{C}(i;T^*) \overset{\cup}{\leftarrow} j$ ($j$ is now a child of $i$), $\mathcal{S}(k;T^*) \overset{\cup}{\leftarrow} (i,j)$ (level $k$ edges) and $A \overset{\cup}{\leftarrow} j$.
```

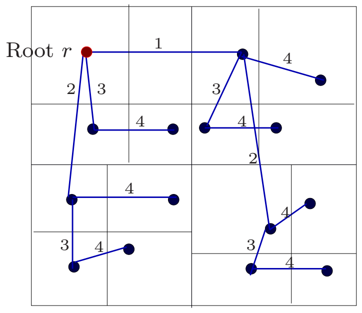

Figure 4: The min. latency tree $T^*(\mathbf{V}_n)$ over 15 nodes with edge level labels placed in square region. See Alg.2.

Definitions: For a rooted tree T and a node i, let $\mathcal{C}(i;T)$ denote the children of i. The *level* $l(e;T)$ of a link e in an aggregation tree T is given by $L_T - t_e$, where t_e is the time at which data is transmitted along link e (time 0 is the start of the aggregation process). Note that the level depends on both the tree structure and the transmission schedule on the tree. Let

$$\mathcal{S}(k;T) := \{e : l(e;T) = k, e \in T\}. \tag{9}$$

be the set of level k edges in tree T. See Fig.3. Let $\mathrm{SP}_l(i,j;\mathbf{V}_n)$ denote the least-energy path[8] between i and j with at most $l \geq 0$ intermediate nodes when the node locations are $\mathbf{V}_n$. For a rectangular region $Q \subset \mathbb{R}^d$ containing a subset of nodes and a reference node i such that $V_i \in Q$, let $\mathcal{B}_1(Q;i), \mathcal{B}_2(Q;i)$ be the two halves when bisected along the coordinate with the largest extent such that $\mathcal{B}_1(Q;i)$ and $\mathcal{B}_2(Q;i)$ have equal number of nodes with $V_i \in \mathcal{B}_1(Q;i)$.

We propose an energy-efficient minimum-latency tree $T^*(\mathbf{V}_n)$ in Algorithm 2, and, prove in Section 3.3 that $T^*(\mathbf{V}_n)$ achieves order-optimal energy consumption for uniformly placed nodes subject to the minimum latency constraint. In Algorithm 2, every added node in the tree picks a new child, as in Algorithm 1, but now the children are chosen based on the node locations. Specifically, in the first iteration, the region of node placement is bisected (with equal number of

[8]Note that the least-energy path depends on the path-loss exponent ν and for larger ν, multi-hop routing is more energy efficient than direct transmissions.

Algorithm 2 Min. lat. tree $T^*(\mathbf{V}_n)$ with order opt. energy

```
Input: Locations of nodes: $\mathbf{V}_n = \{V_1, \ldots, V_n\}$, root node $r$, $Q_n \subset \mathbb{R}^d$: region where the nodes
    are placed. $\mathcal{C}(i;T)$ denotes children of node $i$. $\mathcal{S}(k;T)$ denotes level $k$ edges in $T$. For a
    rectangular region $Q$ and a node $i$ with $V_i \in Q$, let $\mathcal{B}_1(Q;i)$ and $\mathcal{B}_2(Q;i)$ be the two halves with
    $V_i \in \mathcal{B}_1(Q;v)$. For any set $A$, let $A \overset{\cup}{\leftarrow} \{r\}$ denote $A \leftarrow A \cup \{r\}$.
Output: $T^*(\mathbf{V}_n)$.
1: Initialize $A \leftarrow \{r\}$. $R_r \leftarrow Q_n$.
2: for $k = 1, \ldots, \lceil \log_2 n \rceil$ do
3:     $B \leftarrow A$.
4:     for each $i \in B$ do
5:         if $\mathbf{V}_n \cap \mathcal{B}_2(R_i;i) \neq \emptyset$ then
6:             For some node $j$ s.t. $V_j \in \mathcal{B}_2(R_i;i)$, $\mathcal{C}(i;T^*) \overset{\cup}{\leftarrow} j$, $\mathcal{S}(k;T^*) \overset{\cup}{\leftarrow} (i,j)$, $A \overset{\cup}{\leftarrow} \{j\}$, $R_j \leftarrow$
               $\mathcal{B}_2(R_i;i)$ and $R_i \leftarrow \mathcal{B}_1(R_i;i)$.
```

nodes in each half), and the root chooses a child in the other half. The region assigned to the root is now the half-region (where it is located), while the added child node is assigned the other half-region. The subsequent iterations proceed in a similar manner and each node bisects its assigned region into two halves and picks a child in the other half, and updates the assigned regions.

The algorithm 2 considered energy-efficient policy under the minimum latency constraint. We now present the policy π^{AGG} for any given latency constraint in Algorithm 3. The difference between the two cases is that in the latter case, a lower energy consumption is achieved by exploiting the relaxed latency constraint. Intuitively, long-range (direct) communication entails more energy consumption than multi-hop routing, especially when the path-loss exponent ν is large. On the other hand, latency is increased due to multihop routing. The key is to carefully convert some of the long-range links in $T^*(\mathbf{V}_n)$ into multi-hop routes to lower the energy consumption and take advantage of the additional allowed latency.

In Algorithm 3, the regions are bisected and new nodes are chosen as children, as in Algorithm 2. But instead of directly linking the nodes in the two hops, the least-energy route is chosen with at most w_k intermediate routes, where w_k is a fixed weight. The nodes that are already added in this manner are not considered for addition as children in the subsequent iterations. In general, the resulting set of communication links is not a tree, since the least-energy paths constructed in different iterations may share the same set of nodes. But the sum function computation can be carried out on similar lines, as on an aggregation tree. We now relate the weights (w_k) with the latency of the resulting policy π^{AGG} in Algorithm 3.

Proposition 2 (Latency under Algorithm 3) *The aggregation policy π^{AGG} in Algorithm 3 for a given set of weights $\mathbf{w}$ achieves a latency of*

$$L^{\pi^{AGG}}(n) \leq L^*(n) + \sum_{k=0}^{\lceil \log_2 n \rceil - 1} w_k.$$

Proof: There are at most $\lceil \log_2 n \rceil$ iterations and the total delay is

$$\sum_{k=0}^{\lceil \log_2 n \rceil - 1} (1 + w_k) = L^*(n) + \sum_{k=0}^{\lceil \log_2 n \rceil - 1} w_k.$$

Algorithm 3 Latency-energy tradeoff policy $\pi^{\mathrm{AGG}}(\mathbf{V}_n; \mathbf{w})$.

```
Input: Locations of nodes: $\mathbf{V}_n = \{V_1, \ldots, V_n\}$, root node $r$, and set of weights $w_k$ for $k = 0, \ldots, \lceil \log_2 n \rceil - 1$. For a rectangular region $Q$ and node $v \in Q$, let $\mathcal{B}_1(Q; v)$ and $\mathcal{B}_2(Q; v)$ be the two halves with $v \in \mathcal{B}_1(Q; v)$. Let $\mathrm{SP}_l(i, j; \mathbf{V}_n)$ be $l$-hop least-energy path. $Q_n \subset \mathbb{R}^d$: region where the nodes are placed. For any set $A$, let $A \overset{\cup}{\leftarrow} \{r\}$ denote $A \leftarrow A \cup \{r\}$.
Output: $G^{\pi^{\mathrm{AGG}}}$: communication links used by policy $\pi^{\mathrm{AGG}}$.
1: Initialize $A_1, A_2 \leftarrow \{r\}$. $R_r \leftarrow Q_n$.
2: for $k = 0, \ldots, \lceil \log_2 n \rceil - 1$ do
3:   $B \leftarrow A_1$
4:   for each $i \in B$ do
5:     if $(\mathbf{V}_n \cap \mathcal{B}_2(R_i; i)) \setminus A_2 \neq \emptyset$ then
6:       Pick $j$ s.t. $V_j \in \mathcal{B}_2(R_i; i) \setminus A_2$, $A_1 \overset{\cup}{\leftarrow} \{j\}$,
         $G^{\pi^{\mathrm{AGG}}} \overset{\cup}{\leftarrow} \mathrm{SP}_{w_k}(i, j; \mathbf{V}_n)$, $A_2 \overset{\cup}{\leftarrow} \mathrm{SP}_{w_k}(i, j; \mathbf{V}_n)$,
         $R_j \leftarrow \mathcal{B}_2(R_i; i)$ and $R_i \leftarrow \mathcal{B}_1(R_i; i)$.
```

□

Thus, the weights (w_k) can be chosen to satisfy any given latency constraint and we have a policy π^{AGG} for sum function computation given any feasible latency constraint. The analysis of energy consumption under π^{AGG} for a given set of weights is not straightforward to analyze and forms the main result of this paper. This is discussed in the next section.

3.3 Order-Optimality Guarantees

To achieve optimal energy-latency tradeoff according to (6), we choose weights w_k in Algorithm 3, for $k = 0, \ldots, \lceil \log_2 n \rceil - 1$, as

$$w_k = \begin{cases} \lfloor \zeta \delta 2^{k(1/\nu - 1/d)} \rfloor & \text{if } \nu > d, \\ 0 & \text{o.w.} \end{cases} \tag{10}$$

where δ is the additional latency allowed in (6), ν is the path-loss factor for energy consumption in (2) and d is the dimension of Euclidean space where the nodes are placed. The normalizing constant ζ is chosen as

$$\zeta = \begin{cases} 1 - 2^{1/\nu - 1/d}, & \text{if } \nu \geq d, \\ \lceil \log_2 n \rceil^{-1}, & \nu = d\ , \end{cases} \tag{11}$$

so that $\sum_{k=0}^{\lceil \log_2 n \rceil - 1} w_k \leq \delta$. Hence, from Lemma 2, the weights in (10) result in a policy π^{AGG} with latency $L^*(n) + \delta$. We now provide the scaling behavior of optimal energy consumption as well the order-optimality result for π^{AGG}.

Theorem 1 (Energy-Latency Tradeoff) *For a given additional latency constraint $\delta = \delta(n) \geq 0$ and fixed path-loss factor $\nu > 1$ and dimension $d \geq 1$, as the number of nodes $n \to \infty$, the minimum energy consumption for sum function computation satisfies*

$$\mathbb{E}(\mathcal{E}^*(\mathbf{V}_n; \delta)) = \begin{cases} \Theta(n) & \nu < d, \\ O\big(\max\{n, n(\log n)(1 + \frac{\delta}{\log n})^{1-\nu}\}\big) & \nu = d, \\ \Theta\big(\max\{n, n^{\nu/d}(1 + \delta)^{1-\nu}\}\big) & \nu > d, \end{cases}$$

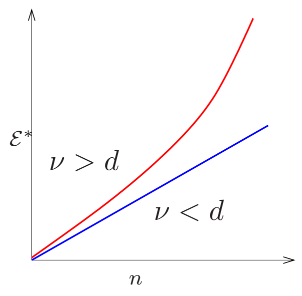

Figure 5: Scaling of minimum total energy $\mathcal{E}^*$ in different regimes of path loss ν and dimension d. See Theorem 1.

where the expectation is over the locations $\mathbf{V}_n$ *of* n *nodes chosen uniformly at random in* $[0, n^{1/d}]^d$ *and is achieved by the policy* π^{AGG} *in Algorithm 3 for weights given by* (10).

Remarks:
*(i)*The policy π^{AGG} in Algorithm 3 thus achieves order-optimal energy consumption under any feasible latency constraint when $\nu \neq d$. For the case $\nu = d$, we show that $\mathbb{E}[\mathcal{E}_n^*(\mathbf{V}_n; \delta)] = \Omega(n)$ while the energy consumption under π^{AGG} is the upper bound in Theorem 1, given by $O(n \log n)$, i.e., the energy consumption is at most only logarithmically worse than the lower bound.
*(ii)*The result of Theorem 1 holds even if the latency constraint is relaxed to an average constraint, i.e.,

$$\mathbb{E}[L^{\pi}(\mathbf{V}_n)] \leq L^*(n) + \delta.$$

From Theorem 1, the energy consumption has different behaviors in the regimes $\nu < d$ and $\nu > d$, as represented in Fig.5, and we discuss this below.
Case $\nu < d$*:* In this regime, the path-loss factor is low, and hence, long-range transmission does not suffer a high penalty over multihop routing. This is also favorable for latency performance and hence, in this regime minimum latency of $\lceil \log_2 n \rceil$ can be achieved with $\Theta(n)$ energy consumption. Note that the aggregation tree with the minimum energy consumption is the minimum spanning tree (MST) and the expected energy consumption for MST under uniform node placement is also $\Theta(n)$. Hence, our policy π^{AGG} achieves both order-optimal energy and minimum latency simultaneously in the regime $\nu < d$.
Case $\nu > d$*:* In this regime, multihop routing is much more favorable over direct transmissions with respect to energy consumption while direct transmissions are favorable for low latency. Hence, both low energy and low latency cannot be achieved simultaneously in this regime. Our policy π^{AGG} achieves order-optimal energy consumption subject to a given latency constraint in this regime. Note that typically, for sensor networks placed in two-dimensional area ($d = 2$) with wireless transmissions ($\nu \in [2, 6]$), this regime is of interest.
Comparison with MST*:* If the minimum spanning tree (MST) is used for aggregation, it results in minimum energy consumption which is $\Theta(n)$ under random node placement. However, the expected latency of aggregation along the MST is at least the depth of the MST and hence, the latency and energy satisfy,

$$\mathbb{E}[L^{\pi^{\text{MST}}}(\mathbf{V}_n)] = \Omega(n^{1/d}), \;\; \mathbb{E}[\mathcal{E}^{\pi^{\text{MST}}}(\mathbf{V}_n)] = \Theta(n). \tag{12}$$

In contrast, under our policy π^{AGG}, we can obtain, when $\nu < d$,

$$\mathbb{E}[L^{\pi^{\text{AGG}}}(\mathbf{V}_n)] = \lceil \log_2 n \rceil, \;\; \mathbb{E}[\mathcal{E}^{\pi^{\text{AGG}}}(\mathbf{V}_n)] = \Theta(n). \tag{13}$$

For the case when $\nu > d$, our policy achieves

$$\mathbb{E}[L^{\pi^{\text{AGG}}}(\mathbf{V}_n)] = \Theta(n^{\frac{\nu/d-1}{\nu-1}}), \;\; \mathbb{E}[\mathcal{E}^{\pi^{\text{AGG}}}(\mathbf{V}_n)] = \Theta(n), \tag{14}$$

by setting $\delta = n^{\frac{\nu/d-1}{\nu-1}}$ in Theorem 1 Thus, our policy π^{AGG} is especially advantageous over using the MST when the path loss ν is small. Moreover, the policy π^{AGG} can be designed based on the latency requirements while the MST cannot be easily modified to satisfy them.

4 General Function Computation

We now extend the latency-energy tradeoff policy to undertake general function computation. Recall that we consider the class of functions of the form

$$\Psi(\mathbf{V}_n, \mathbf{Y}_n) = \sum_{c \in \mathcal{C}(\mathbf{V}_n)} \psi_c((Y_i)_{i \in c}), \tag{15}$$

where $\mathcal{C}(\mathbf{V}_n)$ is the set of *maximal cliques* on a graph $\mathcal{G}(\mathbf{V}_n)$, known as the function dependency graph and the functions ψ_c are clique functions. We consider the case when the graph[9] is either a k-nearest neighbor graph (k-NNG) or the ρ-random geometric graph (ρ-RGG) with threshold radius ρ, where k, ρ are some fixed constants, independent of the number of nodes n.

Note that the function Ψ in (15) now depends on the location of the nodes $\mathbf{V}_n$ which is not the case with the sum function. Hence, the latency-energy analysis has to take this into account. We propose modifications to the latency-energy tradeoff policy π^{AGG} to enable general function computation and then prove its order-optimality for latency-energy tradeoff.

4.1 Preliminaries

The extent of decomposition of function Ψ depends on the *sparsity* (number of edges) of the function dependency graph $\mathcal{G}$. We first make a simple observation that the energy consumption and latency increase with more edges in $\mathcal{G}$.

Proposition 3 (Energy consumption and sparsity of $\mathcal{G}$) *The minimum energy in* (6) *required to compute functions of the form in* (15) *under a fixed additional latency constraint $\delta \geq 0$ with dependency graphs $\mathcal{G}$ and $\mathcal{G}'$ satisfies*

$$\mathcal{E}^*(\mathbf{V}_n; \delta, \mathcal{G}) \geq \mathcal{E}^*(\mathbf{V}_n; \delta, \mathcal{G}'), \quad \textit{when } \mathcal{G} \supset \mathcal{G}'. \tag{16}$$

Proof: Let $\mathcal{C}$ (resp. $\mathcal{C}'$ be the set of maximal cliques in $\mathcal{G}$ (resp. $\mathcal{G}'$). Since $\mathcal{G}' \subset \mathcal{G}$, each new clique $c \in \mathcal{C} \setminus \mathcal{C}'$ replaces a smaller set of cliques $c' \subset c, c' \in \mathcal{C}'$. The latency and energy consumption for any valid policy in computing the clique function $\psi_c((Y_i)_{i \in c})$ is at least that of computing $\sum\limits_{c' \subset c, c' \in \mathcal{C}'} \psi_{c'}(\mathbf{Y}_{c'})$, since this is a further decomposition of ψ_c. Hence, the result. □

[9]In fact, our results hold for a general class of graphs satisfying a certain stabilization property. See [15] for details and examples.

Algorithm 4 Policy $\pi^{\text{CLQ}}(\mathbf{V}_n; \mathbf{w}, \mathcal{C})$ for general functions.

```
Input: Locations of nodes: $\mathbf{V}_n = \{V_1, \ldots, V_n\}$, root node $r$, $\mathcal{C}$: set of maximal cliques of function
    dependency graph $\mathcal{G}(\mathbf{V}_n)$. For each $c \in \mathcal{C}$, $\mathcal{P}(c)$ is the processor (node computing clique function
    $\psi_c$). For any set $A$, let $A \overset{\cup}{\leftarrow} \{r\}$ denote $A \leftarrow A \cup \{r\}$.
Output: $\pi^{\text{CLQ}}$ policy with data forwarding links $F^{\pi^{\text{CLQ}}}$ and aggregation links $G^{\pi^{\text{AGG}}}$.
1: for each clique $c \in \mathcal{C}$ do
2:     For node $i \in c$ with smallest label, $\mathcal{P}(c) \leftarrow i$.
3:     For all nodes $j \in c$, $j \neq i$, $F^{\pi^{\text{CLQ}}} \overset{\cup}{\leftarrow} (j, i)$.
4: Let $l_e$ be the color for edge $e \in F^{\pi^{\text{CLQ}}}$ under proper edge coloring with colors $l = 1, 2, \ldots \Delta + 1$.

5: for $t = 0$ to $\Delta$ do
6:     Send measurements using links in $F^{\pi^{\text{CLQ}}}$ of color $t + 1$.
7: Find sum of clique functions using $\pi^{\text{AGG}}$ from Algorithm 3.
```

Hence, the ability to obtain efficient scaling of latency and energy consumption for function computation depends on the sparsity of the function dependency graph $\mathcal{G}$. The extreme case of a trivial graph ($\mathcal{G} = \emptyset$) is the sum function, analyzed in the previous section, while the other extreme is the complete graph ($\mathcal{G} = K_n$), where there is no decomposition of the function. In the latter case, no in-network computation is possible and all the measurements $\mathbf{Y}_n$ need to be routed to the root via least-energy paths. We have the following scaling in this scenario.

Proposition 4 (Scaling Under No Computation) *The minimum latency and minimum energy (with no latency constraint) for computation of a function with dependency graph K_n satisfies*

$$\mathbb{E}[\mathcal{E}^*(\mathbf{V}_n; \infty, K_n)] = \Theta(n^{1+1/d}), \ \mathbb{E}[L^*(\mathbf{V}_n; K_n)] = \Omega(n).$$

The result on minimum energy follows from the scaling behavior of energy for least-energy path routing to the root under uniform node placement [17]. The latency of function computation is at least n since the root can receive at most one measurement value at each timestep and there is no aggregation of the measurements. Hence, we can expect efficient scaling of energy and latency only in case of computation of functions with sparse dependency graphs $\mathcal{G}$.

Moreover, the energy consumption also depends on the edge lengths of the function dependency graph $\mathcal{G}$. Intuitively, when the graph $\mathcal{G}$ has local edges, the clique functions ψ_c can be computed locally resulting in low energy consumption. This holds for the proximity graphs such as the k-NNG and the ρ-RGG ensure under consideration. We propose policies for latency-energy tradeoff which are efficient for such locally-defined dependency graphs.

4.2 Policy for Latency-Energy Tradeoff

We now extend the policy π^{AGG} in Algorithm 3 for general function computation as a two-stage policy π^{CLQ}. In the first stage known as the data forwarding stage, the clique functions ψ_c are computed locally within each maximal clique $c \in \mathcal{C}$ of the graph $\mathcal{G}$ as follows: a *clique processor* is chosen as a clique member with the smallest label (under arbitrary labeling of nodes) and other clique members communicate their measurements to the processor via direct transmissions. The transmissions are scheduled as follows: the set of forwarding links $F^{\pi^{\text{CLQ}}}$ are assigned colors

$l = 1, 2, \ldots, \Delta + 1$ under a proper edge coloring. At times $t = 0, 1, \ldots, \Delta$, transmissions along links of color $t + 1$ are scheduled simultaneously. In the second stage, the aggregation policy π^{AGG} in Algorithm 3 is used for computing the sum of the clique function values at the processors (and nodes other than processors do not have their own values for aggregation but participate in the process). This is summarized in Algorithm 4.

We obtain the following result for energy-latency tradeoff[10] for general function computation. Let $\Delta(\mathcal{G})$ denote the maximum degree of the function dependency graph $\mathcal{G}$ in (15), which is either the k-NNG or the ρ-RGG, where k and ρ are fixed constants.

Theorem 2 (Energy-Latency Tradeoff) *For a given additional latency constraint $\delta \geq \Delta(\mathcal{G}) + 1$ in* (6)*, the energy consumption for function computation of the form* (15) *with dependency graph $\mathcal{G}$ under the two-stage policy π^{CLQ} satisfies*

$$\mathbb{E}(\mathcal{E}^{\pi^{CLQ}}(\mathbf{V}_n; \delta, \mathcal{G})) = \Theta(\mathbb{E}(\mathcal{E}^*(\mathbf{V}_n; \delta - (\Delta + 1), \emptyset))),$$

where the expectation is over the locations $\mathbf{V}_n$ of n nodes chosen uniformly at random in $[0, n^{1/d}]^d$ *and the right-hand side is the minimum energy consumption for sum function computation under latency constraint of $\delta - (\Delta + 1)$ which is given by Theorem 1.*

Remarks:
(i) The policy π^{CLQ} achieves order-optimal energy consumption for cases $\nu < d, \delta \geq \Delta + 1$ and $\nu > d, \delta \gg \Delta$. This is because the minimum energy consumption $\mathcal{E}^*(\mathbf{V}_n; \delta, \mathcal{G})$ is lower bounded by the minimum energy for sum function computation from Proposition 3. Theorem 1 provides the scaling for minimum energy for sum function computation. Comparing it with the energy under π^{CLQ} policy in Theorem 2, we note that they are both $\Theta(n)$ when $\nu < d$. For the case $\nu > d$, they are still of the same order if the maximum degree $\Delta(\mathcal{G})$ is small compared to additional latency constraint δ.
(ii) The maximum degrees of k-NNG and ρ-RGG satisfy

$$\Delta(k\text{-NNG}) = (c_d + 1)k, \quad \Delta(\rho\text{-RGG}) = \Theta(\frac{\log n}{\log \log n}), \tag{17}$$

where c_d is a constant (depending only on d). See [18, Cor. 3.2.3] and [19, Thm. 6.10]. Hence, for these graphs, π^{CLQ} policy is order-optimal (up to logarithmic factors) for any path-loss factor $\nu \neq d$ and under any additional latency constraint $\delta \geq \Delta(\mathcal{G}) + 1$. The above discussion also implies that the minimum energy for sum function computation and general function computation are of the same order for k-NNG and ρ-RGG dependency graphs. Hence, these functions are amenable to efficient latency-energy tradeoff.
*(iii)*The policy π^{CLQ} can achieve a latency of $\lceil \log_2 n \rceil + \Delta(\mathcal{G}) + 1$. Finding the policy with minimum latency L^* in (5) for general function computation is NP-hard. However, we have $L^* \geq \lceil \log_2 n \rceil$, since the minimum latency cannot be smaller than that required for sum function computation. We ensure that an additional latency constraint of δ is satisfied in (6) by relaxing the constraint as $L \leq \lceil \log_2 n \rceil + \delta$. Since π^{CLQ} can only achieve latencies greater than $\lceil \log_2 n \rceil + \Delta(\mathcal{G}) + 1$, we can only ensure that constraints $\delta \geq \Delta(\mathcal{G}) + 1$ are met.

[10]The latency constraint $L^\pi \leq L^* + \delta$ is required a.s. over the realization of points $\mathbf{V}_n$.

5 Conclusion

In this paper, we considered energy-latency tradeoff for function computation in random networks. While designing optimal tradeoff policies in arbitrary networks is intractable, we proposed simple and easily implementable policies which have order-optimal performance and are relevant in large networks. We analyzed the scaling behavior of energy consumption under a latency constraint for computation and showed that it depends crucially on the path-loss exponent of signal propagation, the dimension of the Euclidean region where the nodes are placed and the extent to which the function is decomposable. For functions which decompose according to cliques of a proximity graph such as the k nearest-neighbor graph or the random geometric graph, efficient tradeoff can be achieved and the energy and latency having optimal scaling behaviors.

This work opens up an array of important and challenging questions which warrant further investigation. While, we considered exact computation of a deterministic function, we expect that relaxing these assumptions will lead to a significant improvement of energy and latency scaling. We assumed single-shot data aggregation. Extensions to the setting of continuous monitoring and collection, where block coding is possible is of interest. We considered a single root node as the destination for the computed function, while in reality different nodes may require different functions to be computed. An extreme case of this scenario is the *belief propagation* (BP) algorithm which requires computation of *maximum a posteriori* (MAP) estimate at each node based on all the measurements, which are drawn from a Markov random field. Considering scenarios between these extremes and designing efficient schemes for energy-latency tradeoff is extremely relevant to many network applications.

A Proof of Lemma 1

We prove by induction on L that the maximum number of vertices in a tree of latency at most L is exactly 2^L. This is clear for $L = 0$ as such a tree must consist of just the root. Now assume $L > 0$ and suppose T is a tree with latency L. Consider the edges that transmit information at the last time step L. Clearly these must transmit to the root r. But the root can only receive information from one child at a time. Thus there is precisely one edge (r, i) along which information is transmitted at time L. Removing the edge (r, i) splits the tree T into two trees T_r and T_i rooted at r and i respectively. For all the data to be received at r in time L, all the data must be received at either r or i by time $L-1$. Thus both T_r and T_i are trees of latency at most $L-1$. By induction T_r and T_i have at most 2^{L-1} vertices. Thus T has at most $2^{L-1} + 2^{L-1} = 2^L$ vertices. Conversely, given two copies of a rooted tree on 2^{L-1} vertices with latency $L-1$, one can construct a tree with latency L on 2^L vertices by joining the roots r, i of these two trees with an edge (r, i), and declaring one of the two roots, say r, to be the root of the resulting tree. The transmission protocol to achieve latency L is simply to follow the protocols on each tree for the first $L-1$ steps, and then transmit all data at i from i to r at time step L.

As any rooted subtree of a tree T has latency at most L_T, it is clear that the minimum latency of any tree on n vertices is $L = \lceil \log_2 n \rceil$, and this can be achieved by taking any rooted subtree of the tree on 2^L vertices constructed above. $\square$

B Proof of Lower Bound in Theorem 1

Note that for $\nu < d$, since the MST has energy $\Theta(n)$, the result follows. For the case $\nu > d$, consider an arbitrary spanning tree with root r. Consider the path P_u from r to u in the tree. Let $R(P_u)$ be the length of P_u, i.e., the sum of the lengths of the edges of P_u. Then with high probability

$$\sum_u R(P_u) \geq \sum_u \|u - r\| \geq cnn^{1/d}. \tag{18}$$

for some constant $c > 0$. Indeed, with high probability, at least one half of the nodes lie at distance at least $\frac{1}{4}n^{1/d}$ from r. Let n_e be the number of paths P_u that go through e, so n_e is the number of vertices below e in the tree. Then

$$\sum R(P_u) = \sum_e R_e n_e.$$

Indeed, $\sum R(P_u)$ counts the length of e exactly n_e times. Now $\mathcal{E}_T = \sum R_e^\nu$, so by Hölder's inequality

$$\Big(\sum R_e^\nu\Big)^{1/\nu}\Big(\sum n_e^{\nu/(\nu-1)}\Big)^{(\nu-1)/\nu} \geq \sum_e R_e n_e \geq cnn^{1/d}. \tag{19}$$

Thus it is enough to find an upper bound on $\sum n_e^{\nu/(\nu-1)}$. If e is at distance i from r then the latency of the tree from e onwards is at most $L^* + \delta - i$. But this means it has at most $2^{L^*+\delta-i} \leq (2n)2^{\delta-i}$ vertices. Hence $n_e \leq n2^{1+\delta-i}$. Also, for each i we have

$$\sum_{e:\mathrm{dist}(e,r)=i} n_e \leq n$$

as each vertex can be counted in at most one n_e with $d(e,r) = i$. Thus

$$\sum_{\mathrm{dist}(e,r)=i} n_e^{\nu/(\nu-1)} = \sum_{\mathrm{dist}(e,r)=i} n_e n_e^{1/(\nu-1)} \leq n(n2^{1+\delta-i})^{1/(\nu-1)}$$

for $i > \delta$, and

$$\sum_{\mathrm{dist}(e,r)=i} n_e^{\nu/(\nu-1)} \leq nn^{1/(\nu-1)}$$

for $i \leq \delta$ as we always have $n_e \leq n$. The first sum is decreasing geometrically in i, so

$$\sum_e n_e^{\nu/(\nu-1)} = (\delta + O(1))nn^{1/(\nu-1)} = O(1+\delta)n^{\nu/(\nu-1)}.$$

Thus by (19),

$$\mathcal{E}_T^{1/\nu}(1+\delta)^{(\nu-1)/\nu}n \geq c'nn^{1/d}.$$

Hence

$$\mathcal{E}_T = \Omega(n^{\nu/d}(1+\delta)^{1-\nu})$$

as required.

The proof that π^{AGG} in Algorithm 3 provides the correct upper bound is given in Appendix D.

C Proof of Theorem 2

For policy π^{CLQ}, the two stages of data forwarding and aggregation do not overlap. Hence, the latency and energy consumption under π^{CLQ} are sum of the respective quantities in the two stages. The aggregation stage uses the π^{AGG} policy and we have results from Theorem 1. We now derive latency and energy scaling laws for the data-forwarding stage under π^{CLQ} policy. Note that these depend on the function dependency graph $\mathcal{G}$.

We claim that the latency in the data forwarding stage of π^{CLQ} is at most $\Delta(\mathcal{G}) + 1$. This is because the forwarding graph is a subgraph of the dependency graph ($F^{\pi^{\text{CLQ}}} \subset \mathcal{G}$) (with directions ignored) since each edge in $\mathcal{G}$ is traversed at most once, under the clique processor selection procedure described in Algorithm 4. Note that any graph with maximum degree Δ has a proper edge coloring using at most $\Delta + 1$ colors (Vizing's theorem) [20]. By scheduling transmissions of a single color in each timestep, we can ensure that each node is either transmitting/receiving from at most one node. The energy consumption is at most the sum of the power-weighted edges of $\mathcal{G}$ and hence,

$$\mathcal{E}(\pi^{\text{CLQ}}; \delta, \mathcal{G}) \leq \sum_{e \in \mathcal{G}} R_e^{\nu} + \mathcal{E}(\pi^{\text{AGG}}; \delta, \emptyset).$$

From [21], for stabilizing graphs $\mathcal{G}$ (which include k-NNG and RGG), we have for uniform node sets $\mathbf{V}_n$,

$$\lim_{n \to \infty} \frac{1}{n} \sum_{e \in \mathcal{G}(\mathbf{V}_n)} R_e^{\nu} \overset{L^2}{=} \zeta < \infty.$$

Hence, $\mathbb{E}[\sum_{e \in \mathcal{G}} R_e^{\nu}] = \Theta(n)$ and we have the result.

D Upper bound construction for Theorem 1

We shall need the following technical lemmas.

Lemma 2 *Let $C > 1$, $d \geq 1$ and $\nu \geq 0$ be constants. Let x be a point of the rectangular parallelepiped $Q = \prod_{i=1}^{d} [0, a_i] \subset \mathbb{R}^d$ of volume $n = \prod a_i$, and bounded aspect ratio, $a_i/a_j \leq C$ for all i, j. Then as $n \to \infty$,*

$$\mathbb{E}(\min_{w \in \mathbf{V}_n} \|w - x\|^{\nu}) = \Theta(1),$$

where the expectation is over all sets $\mathbf{V}_n$ of n points in Q chosen independently and uniformly at random.

The implied constants may depend on C, d and ν, but not on n.

Proof: Let $r > 0$ and set $\mathcal{V}_r$ to be the volume of the set of points in Q that are within distance r of x. Clearly $\mathcal{V}_r$ is at most the volume of a d-dimensional sphere of radius r, i.e., $\mathcal{V}_r \leq cr^d$ for some constant $c = c(d)$. For a lower bound we note that $\mathcal{V}_r \geq 2^{-d} cr^d$ if r is sufficiently small, the worst case being when x is at a corner of Q. However, for a general bound valid for larger r we use the following observation. If $r = \operatorname{diam} Q$, the diameter of Q, then $\mathcal{V}_r$ is the volume of Q, which is just n. If $r \leq \operatorname{diam} Q$ then, by shrinking Q by a linear factor $r/\operatorname{diam} Q$ about x and noting that the resulting set Q' lies inside $Q \cap \mathcal{V}_r$, we see that $\mathcal{V}_r \geq |Q'| = (r/\operatorname{diam} Q)^d n$. As $a_i/a_j \leq C$ for all i, j, we have $\operatorname{diam} Q = \Theta(n^{1/d})$ and hence $\mathcal{V}_r \geq c' r^d$ for some constant $c' = c'(d, C) > 0$.

Using integration by parts we can write

$$\begin{aligned}\mathbb{E}(\min_{w\in\mathbf{V}_n}\|w-x\|^\nu) &= \int_0^{\operatorname{diam} Q} r^\nu(-\tfrac{d}{dr}\mathbb{P}(\min_{w\in\mathbf{V}_n}\|w-x\|\geq r)\,dr,\\ &= \big[-r^\nu\mathbb{P}(\min_{w\in\mathbf{V}_n}\|w-x\|\geq r)\big]_0^{\operatorname{diam} Q}\\ &\quad+\int_0^{\operatorname{diam} Q}\big(\tfrac{d}{dr}r^\nu\big)\mathbb{P}(\min_{w\in\mathbf{V}_n}\|w-x\|\geq r)\,dr\\ &= \int_0^{\operatorname{diam} Q}\nu r^{\nu-1}\mathbb{P}(\min_{w\in\mathbf{V}_n}\|w-x\|\geq r)\,dr\end{aligned}$$

as $\nu > 0$ and (for $n > 0$) there is always a point within distance $\operatorname{diam} Q$ of x. However, $\min_{w\in\mathbf{V}_n}\|w-x\|\geq r$ if and only if there is no point of $\mathbf{V}_n$ in $\mathcal{V}_r$. Thus

$$1-\mathcal{V}_r\leq\mathbb{P}(\min_{w\in\mathbf{V}_n}\|w-x\|\geq r)=(1-\mathcal{V}_r/n)^n\leq\exp(-\mathcal{V}_r).$$

Hence

$$\int_0^{1/c^{1/d}}\nu r^{\nu-1}(1-cr^d)\,dr\leq\mathbb{E}(\min_{w\in\mathbf{V}_n}\|w-x\|^\nu)\leq\int_0^\infty\nu r^{\nu-1}\exp(-c'r^d)\,dr,$$

where we have restricted the range of integration to a constant for the lower bound (which is less than $\operatorname{diam} Q=\Theta(n^{1/d})$ for sufficiently large n), and extended the range on the upper bound. However, both bounds are now positive constants independent of n, although they do depend on d, ν, and C. Thus $\mathbb{E}(\min_w\|w-x\|^\nu)=\Theta(1)$ as required. □

Given a set $\mathbf{V}_n\subseteq\mathbb{R}^d$ and two points $u,v\in\mathbb{R}^d$, a $\mathbf{V}_n$-path of length k from u to v is a sequence $ux_1x_2\dots x_{k-1}v$ with $u_1,\dots,u_{k-1}\in\mathbf{V}_n$. As before, the *energy* of a path $P=u_0u_1\dots u_k$ is

$$E_P=\sum_{i=1}^k\|u_i-u_{i-1}\|^\nu.$$

Lemma 3 *Let u and w be points in $Q=[0,n^{1/d}]^d$ at distance $R_{uw}=\|u-w\|$ from each other and let $\mathbf{V}_n$ be a set of n points in Q chosen uniformly and independently. Let $E_k(u,v;\mathbf{V}_n)$ be the minimal energy of a $\mathbf{V}_n$-path of length at most k from u to v. Then*

$$\mathbb{E}\big(E_k(u,v;\mathbf{V}_n)\big)=O\big(\max\{k(R_{uw}/k)^\nu,R_{uw}+1\}\big).$$

Proof: Let $x_1,\dots,x_{k-1}$ be $k-1$ subdivision points on the line segment from u to v, equally spaced so that this line segment is divided into k equal segments. Pick u_i to be the point of $\mathbf{V}_n$ closest to x_i, and also write $u_0=x_0=u$ and $u_k=x_k=v$. Then, after possibly removing loops and repeated vertices, $u_0u_1\dots u_k$ gives a suitable path from u to v of energy at most $\sum_{i=1}^k\|u_i-u_{i-1}\|^\nu$. Now

$$\begin{aligned}\|u_i-u_{i-1}\|^\nu &\leq 3^\nu(\tfrac{1}{3}(\|u_i-x_i\|+\|u_{i-1}-x_{i-1}\|+\|x_i-x_{i-1}\|))^\nu\\ &\leq 3^\nu\max\{\|u_i-x_i\|^\nu,\|u_{i-1}-x_{i-1}\|^\nu,\|x_i-x_{i-1}\|^\nu\}.\end{aligned}$$

Hence the energy of the path is at most $O(k(R_{uw}/k)^\nu)+O(\sum\|u_i-P\|^\nu)$. By Lemma 2, $\mathbb{E}(\sum\|u_i-x_i\|^\nu)=k\Theta(1)$. If $k\leq R_{uw}$ then the expected energy of the path is at most $O(k(R_{uw}/k)^\nu)$, as required. If $k > \ell_{uw}$, then we perform the same construction replacing k with $\lceil R_{uw} \rceil$ to obtain a path of expected weight $O(R_{uw} + 1)$. $\square$

Given ν, d, δ, our aim is construct a spanning tree T with latency at most $L^*(n) + \delta$ and small energy $E_T = \sum R_e^\nu$. The tree will be formed from a minimal latency tree T_L with latency $L = L^*(n) = \lceil \log_2 n \rceil$. Some edges will be subdivided and not all vertices will be present. Some other minor changes to the tree may also be made. The edges of T_L of level i will be subdivided at most

$$s_i := \lfloor c_1 \delta 2^{i(1/\nu - 1/d)} \rfloor \tag{20}$$

times when $\nu \geq d$, where $c_1 = 1 - 2^{1/\nu - 1/d}$ (or $c_1 = 1/\lceil \log_2 n \rceil$ if $\nu = d$) is a normalizing factor chosen so that $\sum_i s_i \leq \delta$. For $\nu < d$ we shall take $s_i = 0$ as subdividing is not necessary. We will of course need to prune some leaves later so that the tree has n vertices. It is clear that the latency of T_L with level i edges subdivided at most s_i times is at most $L^* + \sum s_i \leq L^*(n) + \delta$. Indeed we just replace the single time step $L - i$ with $1 + s_i$ time steps, during which data is transmitted along the (at most) $1 + s_i$ edges of each subdivided level i edge of T_L.

Our strategy is as follows. Given a rectangular d dimensional region Q, containing n points, one of which is chosen as a root vertex v, we pick a coordinate, say x_1, in which Q has largest extent and subdivide $Q = Q_0 \cup Q_1$ so that half of the points are on each side.

The root vertex v will be in one half, say Q_0. Choose a minimal weight path P from v with at most s_0 subdivisions joining v to a vertex v_1 in Q_1. Now inductively repeat the construction within Q_0 and Q_1 using $v_0 = v$ and v_1 as the corresponding root vertices. At the next stage, each of Q_0 and Q_1 is subdivided as above, say $Q_0 = Q_{00} \cup Q_{01}$ and $Q_1 = Q_{10} \cup Q_{11}$. Assume w.l.o.g. that $v_0 \in Q_{00}$, $v_1 \in Q_{10}$. Pick a vertex $v_{01} \in Q_{01}$ that has not already been used and join with a path with at most s_1 subdivisions to $v_{00} = v_0$ inside Q_0. Similarly join some $v_{11} \in Q_{11}$ to $v_{10} = v_1$ inside Q_1. Repeat this process. At each stage, choose the $v_{\dots 1}$ to be any vertex in the other half of the appropriate region that has not already been used on any path so far. If no such vertex exists then the construction within this half terminates as we have exhausted all the vertices.

There are two main problems with this approach, both concerning the applicability of induction to the process. The n points in Q were chosen to be *uniform* and *independent*. If care is not taken with the division of Q into Q_0 and Q_1 then the points in Q_0 and Q_1 will not be uniform. For example, it is *not* sufficient to order the points by the x_1-coordinate and just take some subdivision between the $(n/2)$th and $(n/2 + 1)$st points. Neither will the volumes of Q_0 and Q_1 be exactly $n/2$, although this is a relatively minor problem as we shall see later. The second, and more serious problem is that after constructing the path from v_0 to v_1, some vertices in Q_0 and Q_1 have now been used. the remaining points are now *not* independent. This second point will be dealt with by allowing the subdividing vertices to be reused in subsequent steps, i.e., by allowing subdividing paths to intersect. The resulting graph will then not be a tree, but we shall show that it will still have the same latency and can be modified to form a tree with no increase in latency. To solve the first problem mentioned above we shall use the following lemma.

Lemma 4 *Take n random points independently and uniformly in $[0, 1]$ and order them by value as $0 \leq x_1 \leq x_2 \leq \cdots \leq x_n \leq 1$. Then x_{r+1} has mean $\frac{r+1}{n+1}$ and variance $\frac{(r+1)(n-r)}{(n+1)^2(n+2)}$. Moreover, conditioned on the value of x_{r+1}, the set of values $\{x_1, \dots, x_r\}$ can be obtained by taking r random points independently and uniformly distributed in $[0, x_{r+1}]$.*

Proof: The probability that x_{r+1} lies in a small interval $[x, x + dx]$ is given as the probability that *some* point is in the interval times the probability that there are exactly r of the remaining

$n-1$ points in $[0,x]$. Thus the pdf of x_{r+1} is $n\binom{n-1}{r}x^r(1-x)^{n-r-1}$. This is just the pdf of the beta distribution with parameters $r+1$ and $n-r$. The mean and variance can be obtained by using standard results on the beta distribution, or by a straightforward calculation. For the conditioning result, imagine the points are located in small bins of size dx, chosen so that it is very unlikely that two points appear in the same bin. Conditioning on x_{r+1} lying in the bin $[x, x+dx]$ is equivalent to conditioning on one point lying in this interval, r points lying in earlier bins, and $n-r-1$ points lying in later bins. However and valid assignment of points to bins is equally likely, so the conditional distribution of the first r points is equivalent to choosing r points uniformly and independently in $[0,x]$. □

Corollary 1 *If $r = n/2 + O(1)$ in Lemma 4 and $c = \frac{1}{2} + O(1/n)$, then for $\nu > 0$*

$$\mathbb{E}(\max\{x_{r+1}/c, 1\}^\nu) = 1 + O(1/\sqrt{n}).$$

Proof: Note that x_{r+1}/c is bounded between 0 and $2+\varepsilon$ for large n, so $\max\{x_{r+1}/c, 1\}^\nu \leq 1 + K|x_{r+1} - c|$ for some constant $K = K(\nu) > 0$. Now

$$\mathbb{E}(|x_{r+1} - c|)^2 \leq \mathbb{E}(|x_{r+1} - c|^2) = \mathrm{Var}(x_{r+1}) + (\mathbb{E}(x_{r+1}) - c)^2.$$

However, by Lemma 4, $\mathbb{E}(x_{r+1}) - c = O(1/n)$ and $\mathrm{Var}(x_{r+1}) = O(1/n)$. The result follows. □

Proof of the upper bound for Theorem 1: We follow the strategy outlined above. At step i have n_i vertices, $\lfloor n/2^i \rfloor \leq n_i \leq \lceil n/2^i \rceil$, chosen uniformly at random from a rectangular parallelepiped $Q = \prod[0, a_j]$ of volume n_i and aspect ratio $a_j/a_k \leq 3$ for all j, k. We wish to construct a spanning tree of latency at most $L_i = L^*(n) - i + \sum_{j \geq i} s_j$ of expected energy at most M_i; the value of M_i to be determined.

If $n_i = 1$ then there is nothing to do, so assume $n_i > 1$. Pick a coordinate x_j with maximum value of a_j. W.l.o.g. $j = 1$. Order the n_i points by their x_1-coordinates, which are uniform and independent in $[0, a_1]$. Let $x_1(r)$ be the value of the rth x_1-coordinate. Define Q_0 to be the subset of points of Q with x_1-coordinate less than $x_1(n_{i+1}+1)$, where $n_{i+1} = \lfloor n_i/2 \rfloor \in [\lfloor n/2^{i+1} \rfloor, \lceil n/2^{i+1} \rceil]$. Let Q_1 to be the subset of points of Q with x_1-coordinate more than $x_1(n_{i+1})$. Note that Q_0 and Q_1 overlap, however their intersection contains no points of $\mathbf{V}_n$. Also note that (with probability 1), there are n_{i+1} points in Q_0 and $n_i - n_{i+1} = \lceil n_i/2 \rceil \in [\lfloor n/2^{i+1} \rfloor, \lceil n/2^{i+1} \rceil]$ points in Q_1. By Lemma 4, conditioned on the shape of Q_0, say, the points in Q_0 are uniformly distributed inside Q_0. (There is however a dependency of these points on the shape of Q_1.) If the root lies in Q_0 then we use this as the root of Q_0, otherwise we pick any point in $\mathbf{V}_n \cap Q_0$ as the root. Shrink Q_0 in the x_1-direction by a factor $X_0 = (x_1(n_{i+1}+1)/a_1)/(n_{i+1}/n_i)$ to obtain a set Q_0' with volume exactly n_{i+1}. Now construct a spanning tree of latency at most L_{i+1} on the points in Q_0'. Let E_0 be the energy of this tree. Now scaling by a factor of X_0 we obtain a tree T_0 in Q_0 of expected energy at most $\max\{1, X_0\}^\nu E_0$, all lengths having increased by a factor of at most $\max\{1, X_0\}$. Note that X_0 and E_0 are independent, so the expected value of the energy of this tree is at most $\mathbb{E}(\max\{1, X_0\}^\nu)\mathbb{E}(E_0)$ which is at most $(1 + C/\sqrt{n_i})\mathbb{E}(E_0)$ by Corollary 1. Since a_1 was maximal and the x_1-extent of Q_0' is $a_1 n_{i+1}/n_i \geq a_1/3$, the aspect ratios of Q_0' are again at most 3. The points in Q_0 are distributed uniformly at random and the volume of Q_0' is n_{i+1}. Thus by reverse induction on i, $\mathbb{E}(E_0) \leq M_{i+1}$. A similar argument also applies to Q_1, so we obtain two trees with expected total energy of at most $(1 + C/\sqrt{n_i})M_{i+1}$. By Lemma 3 we can find a path from the root

v_0 of Q_0 to the root v_1 of Q_1 of length at most $1+s_i$ and expected energy at most

$$C' \max\{n_i^{\nu/d}(1+s_i)^{1-\nu}, n_i^{1/d}+1\}$$

Note that this path will reuse vertices of the trees T_0 and T_1, so that we may obtain a graph G with cycles. However, the latency of G is still at most L_i. Indeed, in the first $L_{i+1} = L^*(n) - (i+1) + \sum_{j\geq i+1} s_j$ steps we transmit the data to v_0 and v_1 in T_0 and T_1 separately. Then in the last $1+s_i$ steps we transmit the data along the $v_0 - v_1$ path P to whichever of these vertices was the root of Q. If one insists on having a tree then one can modify G slightly as follows. Any vertex of T_0 or T_1 that lies in P holds the data rather than transmitting it on during the first L_{i+1} steps. Tree edges from these vertices towards the roots v_0 and v_1 are deleted from T_0 and T_1 as these edges are no longer used. The trees T_0 and T_1 now become a forest of possibly many trees. All the data after the first L_{i+1} steps is located at vertices that are roots of all the individual trees in this forest. These vertices all lie on P. The path P then joins all the roots of these small trees to form one tree and data can be swept up along P in the last $1+s_i$ time steps.

It now remains to bound the expected energy of this tree. We have an upper bound given by the energies of T_0, T_1, and P of

$$M_i := 2(1+C/\sqrt{\tilde{n}_i})M_{i+1} + C' \max\{n_i^{\nu/d}(1+s_i)^{1-\nu}, n_i^{1/d}+1\} \tag{21}$$

where in order to maximize the expression we take $\tilde{n}_i = \lfloor n/2^i \rfloor$ and $n_i = \lceil n/2^i \rceil$. Note $M_i = 0$ for $i \geq \log_2 n$ as then $n_i \leq 1$. Inductively substituting the bound in (21) for M_{i+1} gives

$$M_i \leq \Big(\prod_{j>i}(1+C/\sqrt{\tilde{n}_j})\Big) \sum_{j\geq i} C' 2^{j-i} \max\{n_j^{\nu/d}(1+s_j)^{1-\nu}, n_j^{1/d}+1\}$$

Now the $\tilde{n}_j$ increase exponentially as i decreases from $\log_2 n$, thus the product $\prod_{j>i}(1+C/\sqrt{\tilde{n}_j})$ can be bounded independently of n and i. Hence

$$\begin{aligned} M_0 &\leq C'' \sum_{i=0}^{\log_2 n} 2^i \max\{(n/2^i)^{\nu/d}(1+s_i)^{1-\nu}, (n/2^i)^{1/d}+1\} \\ &\leq C'' \sum_{i=0}^{\log_2 n} 2^i (n/2^i)^{\nu/d}(1+s_i)^{1-\nu} + C'' \sum_{i=0}^{\log_2 n} 2^i((n/2^i)^{1/d}+1) \end{aligned} \tag{22}$$

where C'' absorbs this product and also any factors that arise from the difference between n_j and $n/2^j$. The second sum in (22) is geometrically increasing for $d > 1$ so has sum $O(n)$. Assume $\nu > d$. Then $s_i = \Theta(\delta 2^{i(1/\nu - 1/d)})$ and $1 + (1/\nu - 1/d)(1-\nu) - \nu/d = (1/\nu - 1/d) < 0$. Thus the first sum in (22) is decreasing geometrically when $s_i > 0$, and also decreasing geometrically at an even faster rate when $s_i = 0$. Hence this sum gives $O(n^{\nu/d}(1+s_0)^{1-\nu})$. Thus

$$M_0 = O(\max\{n, n^{\nu/d}(1+\delta)^{1-\nu}\}).$$

for $\nu > d$. For $\nu = d > 1$, $s_i = s_0 = \Theta(\delta/\log n)$ is constant. Thus the terms in the first sum are equal and add up to $(\log n) O(n(1+s_0)^{1-\nu})$. Thus

$$M_0 = O(\max\{n, n(\log n)(1+\delta/\log n)^{1-\nu}\}).$$

for $\nu = d > 1$. Finally, for $\nu < d$, $s_i = 0$ and the first sum is increasing geometrically. This sum is then $O(n)$ and so

$$M_0 = O(n)$$

for $1 < \nu < d$. □

References

[1] A. Anandkumar, J. Yukich, L. Tong, and A. Swami, "Energy Scaling Laws for Distributed Inference in Random Networks," *IEEE J. Selec. Area Comm.*, vol. 27, no. 7, pp. 1203–1217, Sept. 2009.

[2] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," in *Proc. of the 5th ACM/IEEE International Conference on Mobile Computing and Networking*, Aug. 1999, pp. 263–270.

[3] G. Pottie and W. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, vol. 43, pp. 51–58, May 2000.

[4] A. Giridhar and P. Kumar, "Computing and Communicating Functions over Sensor Networks," *IEEE JSAC*, vol. 23, no. 4, pp. 755–764, 2005.

[5] R. Ravi, "Rapid Rumor Ramification: Approximating the Minimum Broadcast Time," in *IEEEE Symposium on Foundations of Computer Science*, vol. 35, 1994, pp. 202–202.

[6] E. Brosh, A. Levin, and Y. Shavitt, "Approximation and heuristic algorithms for minimum-delay application-layer multicast trees," *IEEE/ACM Tran. on Networking*, vol. 15, no. 2, pp. 473–484, 2007.

[7] S. Lindsey, C. Raghavendra, and K. Sivalingam, "Data Gathering in Sensor Networks Using the Energy*Delay Metric," in *Proc. of International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobil Computing*, CA, April 2001.

[8] M. Zorzi and R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: energy and latency performance," *IEEE transactions on Mobile Computing*, pp. 349–365, 2003.

[9] Y. Yu, B. Krishnamachari, and V. Prasanna, "Energy-latency tradeoffs for data gathering in wireless sensor networks," in *Proc. of IEEE INFOCOM*, 2004.

[10] T. Moscibroda, P. von Rickenbach, and R. Wattenhofer, "Analyzing the Energy-Latency Trade-Off During the Deployment of Sensor Networks," in *Proc. of IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[11] Q. Zhao and L. Tong, "Energy Efficiency of Large-Scale Wireless Networks: Proactive vs. Reactive Networking," *IEEE JSAC Special Issue on Advances in Military Wireless Communications*, May 2005.

[12] A. Anandkumar, L. Tong, A. Swami, and A. Ephremides, "Minimum Cost Data Aggregation with Localized Processing for Statistical Inference," in *Proc. of INFOCOM*, Phoenix, USA, April 2008, pp. 780–788.

[13] S. Yi, Y. Pei, and S. Kalyanaraman, "On the capacity improvement of ad hoc wireless networks using directional antennas," in *Proc. of ACM Intl. symposium on Mobile ad hoc networking & computing*, 2003, pp. 108–116.

[14] C. Shen, Z. Huang, and C. Jaikaeo, "Directional broadcast for mobile ad hoc networks with percolation theory," *IEEE Tran. on mobile computing*, pp. 317–332, 2006.

[15] M. Penrose and J. Yukich, “Central Limit Theorems For Some Graphs In Computational Geometry,” *Annals of Applied Probability*, vol. 11, no. 4, pp. 1005–1041, 2001.

[16] A. Anandkumar, L. Tong, A. Swami, and A. Ephremides, “Minimum Cost Data Aggregation with Localized Processing for Statistical Inference,” in *Proc. of IEEE INFOCOM*, Phoenix, USA, Apr. 2008, pp. 780–788.

[17] F. Baccelli and B. Blaszczyszyn, “Stochastic Geometry and Wireless Networks,” *Foundations and Trends in Networking*, 2009.

[18] G. Miller, S. Teng, W. Thurston, and S. Vavasis, “Separators for Sphere-Packings and Nearest Neighbor Graphs,” *J. of the ACM*, vol. 44, no. 1, 1997.

[19] M. Penrose, *Random Geometric Graphs.* Oxford University Press, 2003.

[20] D. West, *Introduction to graph theory.* Upper Saddle River, NJ: Prentice Hall, 2001.

[21] M. Penrose and J. Yukich, “Weak Laws Of Large Numbers In Geometric Probability,” *Annals of Applied Probability*, vol. 13, no. 1, pp. 277–303, 2003.