Time and Energy Complexity of Distributed Computation of a Class of Functions in Wireless Sensor Networks*

Nilesh Khude[1], Anurag Kumar[2], Aditya Karnik [1]

***Abstract*— We consider a scenario in which a wireless sensor network is formed by randomly deploying n sensors to measure some spatial function over a field, with the objective of computing a function of the measurements and communicating it to an operator station. We restrict ourselves to the class of type-threshold functions (as defined in [2]), of which $\max$, $\min$, and indicator functions are important examples; our discussions are couched in terms of the $\max$ function. We view the problem as one of message passing distributed computation over a geometric random graph. The network is assumed to be synchronous; the sensors synchronously measure values, and then collaborate to compute and deliver the function computed with these values to the operator station. Computation algorithms differ in (i) the communication topology assumed, and (ii) the messages that the nodes need to exchange in order to carry out the computation. The focus of our paper is to establish (in probability) scaling laws for the time and energy complexity of the distributed function computation over random wireless networks, under the assumption of centralised contention-free scheduling of packet transmissions. Firstly, without any constraint on the computation algorithm, we establish scaling laws for the computation time and energy expenditure for one time maximum computation. We show that, for an optimal algorithm, the computation time and energy expenditure scale, respectively, as $\Theta\left(\sqrt{\frac{n}{\log n}}\right)$ and $\Theta(n)$ asymptotically as the number of sensors $n \to \infty$. Secondly, we analyze the performance of three specific computation algorithms that may be used in specific practical situations, namely, the Tree algorithm, Multi-Hop transmission, and the Ripple algorithm (a type of gossip algorithm), and obtain scaling laws for the computation time and energy expenditure as $n \to \infty$. In particular we show that the computation time for these algorithms scales as $\Theta\left(\sqrt{n \log n}\right)$, $\Theta(n)$ and $\Theta\left(\sqrt{n \log n}\right)$, respectively; whereas the energy expended scales as $\Theta(n)$, $\Theta\left(n\sqrt{\frac{n}{\log n}}\right)$ and $\Theta\left(n\sqrt{n \log n}\right)$, respectively. Finally, simulation results are provided to show that our analysis indeed captures the correct scaling; the simulations also yield estimates of the constant multipliers in the scaling laws. Our analyses throughout assume a centralized optimal scheduler and hence our results can be viewed as providing bounds for the performance with practical distributed schedulers.**

Keywords: distributed maximum computation, scaling laws for sensor networks

*This work was supported by the Indo-French Centre for Promotion of Advanced Research (IFCPAR) under research contract No. 2900-IT. This is a revised version of a paper that appeared in IEEE Infocom 2005 ([9]).

[1] This work was done when these authors were with ECE Department, Indian Institute of Science, Bangalore, INDIA

[2]ECE Department, Indian Institute of Science, Bangalore, INDIA

I. Introduction

A wireless sensor network is formed by a set of miniature smart sensor devices, each equipped with a digital wireless transceiver, that are deployed in an ad hoc fashion and cooperate in sensing the environment, in computing some quantity of global interest, and in transporting this to a designated "base station" node (for a survey see [1]). Sensor nodes have limited, and in many cases, irreplaceable power sources. Power consumption occurs due to radio transmission, reception, sensing and computing, typically in decreasing order. As a node spends the maximum energy in communication, it is desirable to have local interactions between the sensors to process the data *in* the network rather than to transmit the raw data to the base station. This is because, by reducing the number of packets that need to be transported in the network, in-network computations reduce the packet transport load and thus increase the lifetime of the network. In this paper we focus on the distributed computation approach for sensor information processing.

The work reported in this paper is in the context of the following model. There are n sensor nodes distributed independently and uniformly over a 2-dimensional field $\mathcal{A}$. It is assumed that time is slotted and the sensors are synchronised at slot boundaries. The sensors synchronously sample the environment variable, e.g., temperature. The measurements are assumed to be quantized and take values in a given *finite set*, $\mathcal{V}$. At sampling instant k, each sensor measures a value, yielding a vector of values $\mathbf{v}(k) = (v_1(k), v_2(k), \cdots, v_n(k))$. The objective is to collaboratively compute and deliver $\max\{v_1(k), v_2(k), \cdots, v_n(k)\}$ to an operator station, for each such vector of sampled values. See [4] where the need for a distributed maximum computation arises as a part of a distributed self-tuning algorithm for the optimal operation of a sensor network. If the sensors calculate local maxima while routing the values to the operator station, we can reduce the traffic in the network and thereby decrease the computation delay and increase the network lifetime. In the case of the function $\max$, this is possible because the maximum function is insensitive to the order of computation and can be calculated recursively by using partial results obtained by using arbitrary subsets of the data, i.e., $\max\{a, b, c, d, e\}$ can be calculated as $\max\{\max\{a, b\}, \max\{c, \max\{d, e\}\}\}$.

Although $\max$ will be a convenient example for us to discuss throughout the paper, the class of functions our algorithms and analyses cover is wider. It includes the class of type-threshold

functions, as defined in [2][1]. We also consider a gossip algorithm that we call *Ripple* that works only for the subclass of the above functions whose result does not change if the elements in the argument vector $\mathbf{v}$ are repeated during the computation. Since, for example, $\max\{a,b,c\} = \max\{a,b,a,c,b\}$, the Ripple algorithm applies to the computation of $\max$, $\min$ or set union and set intersection.

We adopt the message passing distributed computing model. The sensors communicate by sending packets to each other and then performing computations based on the received data and the partial results they already have. The computation algorithms we consider differ in the way the computations are organised, and hence in the message transmissions that are required to carry out the task. When successive results for several sampled values need to be computed then separate pipelined computations are performed for each vector of sampled values. Thus, we do not exploit block computation, as has been done in [2]. We assume a centralized optimal scheduler, which schedules maximal independent sets of links in the network. Thus, there are no collisions in the model.

The following is a summary of our contributions in this paper: All our results are of the nature of providing asymptotic scaling laws (that hold in the "in probability" sense) as the number of nodes $n \to \infty$. As has been established in [7] and [6], to maintain the connectivity of the network, the transmission range $r(n)$ must scale as $\Theta\left(\sqrt{\frac{\log n}{n}}\right)$[2]. Adopting these results, and under our assumptions, we establish that the time required for one computation (e.g., initiated by a query) by an optimal algorithm is $\Theta\left(\sqrt{\frac{n}{\log n}}\right)$. The minimum energy expended in the network during a computation is $\Theta(n)$. All these orders are tight bounds in the sense that there exist (centralized) algorithms that achieve these orders. We also analyse the performance (scaling orders with n, of the single computation time and energy expended) of some candidate computation algorithms, thus providing a comparison between them. We consider the Tree, Multi-Hop and Ripple algorithms, and obtain scaling laws for the computation time and energy expenditure as $n \to \infty$. In particular we show that the computation time for these algorithms scale as $\Theta\left(\sqrt{n \log n}\right)$, $\Theta(n)$ and $\Theta\left(\sqrt{n \log n}\right)$, respectively, whereas the energy expended scales as $\Theta(n)$, $\Theta\left(n\sqrt{\frac{n}{\log n}}\right)$ and $\Theta\left(n\sqrt{n \log n}\right)$, respectively. The Tree algorithm outperforms the others in terms of the computation time and energy expediture, and is suitable for data aggregation in sensor networks, but requires the maintenance of the tree. The Multi-Hop algorithm preserves the entire data at the operator station, and, hence, imposes no restrictions on the function to be computed. The Ripple algorithm, though inefficient compared to the other two algorithms, has the advantage of being completely distributed, not requiring any organization in network; the Ripple algorithm also provides the result of the computation to every node. As stated above, all the analyses assume a centralized, collision-free medium access scheduler. Thus the scaling orders we obtain can be viewed as lower bounds when some practical distributed medium access protocol is implemented. Finally, we provide the results of a simulation study of the three algorithms; these results confirm our scaling results, and also provide estimates of the preconstants.

The work we report in this paper is closely related to the one presented in [2]. We will discuss the relationships after formally presenting our distributed computation model in Section II. We will then discuss some background results in Section III. In Section IV, we obtain the optimal order expressions for the performance measures. The performance of some algorithms is analyzed in Section V. Simulation results are presented in Section VI. We conclude the paper in Section VII.

II. The Model and Performance Measures

We consider n sensors deployed in a circular or square field. A sensor located at the coordinate $\mathbf{X}$ measures the value of some spatial function (say, temperature) $f(\mathbf{X})$. We are interested in obtaining the maximum of the measured values and communicating the maximum to an operator station located at the centre of the field.

Network Model: The two dimensional field in which the n sensors are located is denoted by $\mathcal{A}$. The sensor network is characterized by an indexed set of sensor locations $\mathcal{S}$; sensor i has location $\mathbf{X}_i$, $\mathbf{X}_i \in \mathcal{A}$, $1 \leq i \leq n$. The network $\mathcal{S}$ is a random vector $(\mathbf{X}_1, \ldots, \mathbf{X}_n) \in \mathcal{A}^n$ where the $\mathbf{X}_i$s are i.i.d. random variables, each uniformly distributed over $\mathcal{A}$. The random experiment of deploying a network of sensors is characterized by the probability space $\Sigma^n := (\mathcal{A}^n, \mathcal{F}^n, \mathcal{P}^n)$ where $\mathcal{A}^n$ is the sample space, $\mathcal{F}^n$ is the event space (a Borel field) and $\mathcal{P}^n$ is the probability measure. We index the whole experiment by n, the number of nodes deployed in the field. As n increases, we get a sequence of experiments. We wish to study the asymptotics of certain performance measures as $n \to \infty$.

All radio communication is over a common channel and any radio transceiver can either transmit or receive at a time. The *transmission range* of the sensors is fixed for fixed n and is denoted by $r(n)$. If any two sensors are within a distance $r(n)$ of each other then there is a bidirectional link between them. Thus, the neighbours of a node are nodes within a distance $r(n)$ from that node. The form of $r(n)$ we use follows from the results of [6].

Definition 2.1: Given a network realisation $\mathcal{S}$, the graph $G(\mathcal{S})$ is formed by the n nodes at the locations defined by $\mathcal{S}$, with links joining the nodes that are separated by a distance not greater than $r(n)$. Thus, we have a random graph, denoted by G.

[1]The type of a vector of observations $\mathbf{v} = (v_1, v_2, \cdots, v_n) \in \mathcal{V}^n$ is a $|\mathcal{V}|$ dimensional vector. For an element $v \in \mathcal{V}$ the type of the given observation $\mathbf{v}$ is given by $\tau_v(\mathbf{v}) = \sum_{i=1}^{n} I_{\{v_i = v\}}$, i.e., the number of times the value v occurs in $\mathbf{v}$. The functions that we are concerned with are such that there exist certain thresholds $t_v, v \in \mathcal{V}$, and the result of evaluating the function on the observations $\mathbf{v}$ depends on the observations only via $\min\{\tau_v(\mathbf{v}), t_v\}, v \in \mathcal{V}$. It follows that the number bits required to represent the computed value of the function is bounded even as n increases.

[2]A positive function $f(n) = O(g(n))$ implies that there exist, positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$. A positive function $f(n) = \Omega(g(n))$ implies that there exist, positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$. We say that the function $f(n) = \Theta(g(n))$, if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, i.e. there exist positive constants c_1, c_2 and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ ([3]). These notations are used here in the following probabilistic sense : we say $f(n) = \Theta(g(n))$ if and only if there exist positive constants c_1 and c_2 such that $\lim_{n\to\infty} P(0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)) = 1$

Interference Model: Let $|\mathbf{X}_i - \mathbf{X}_j|$ denote the Euclidean distance between the nodes i and j. We adopt the *protocol model* which defines the interference constraints as follows.

Definition 2.2 (Gupta and Kumar [7]): Protocol Model of Interference: When node i transmits to node j (i.e., $i \rightarrow j$), then the transmission is successful if

1) $|\mathbf{X}_i - \mathbf{X}_j| \leq r(n)$ and
2) For every node k that transmits simultaneously, $|\mathbf{X}_k - \mathbf{X}_j| \geq (1+\Delta)r(n)$ for some fixed $\Delta > 0$.

Distributed Computation Model: We work with the model of *message passing* distributed computation. Nodes explicitly send packets to each other, and do not exploit any extra information available on the wireless medium by way of listening to the other nodes' transmissions or to collisions. Nodes perform computations based only on the packets that are explicitly sent to them. This necessitates that to complete a computation, each node must *influence* the computation. As a simple example, suppose there are three collocated sensors with values a, b, and c, such that $a > b > c$, and a collocated operator station. Suppose also that a is the maximum element of the set in which these values are taken. Then, evidently, if the operator station hears that a station has the value a it can declare the $\max$ without waiting to hear from any other node. In our message passing computation model, however, we require that the operator station hears at least once from all the nodes, before declaring that the computation is complete. Note that this does not imply that the operator station must explicitly receive each value, only that the computation it receives must have been *influenced* by every sensor's value. Further, when the computations are performed for several set of measurements, the computation of the maximum for each set of measured values is carried out separately, and block computation is not exploited as in [2].

Formally, suppose that the result delivered to the operator station is $y = \max_{1 \leq i \leq n} \ v_i$, where v_i is the measured value at node i and is obtained as $y = \max\{y_1, y_2, \cdots, y_m\}$, with $y_i = \max_{j \in I_i} \ v_j$, where, for $1 \leq i \leq m$, $I_i \subset \{1, 2, 3, \cdots, n\}$. Now, even though $y = y_j$ for some particular $j, 1 \leq j \leq m$ (i.e., the maximum is determined by the subset of sensors I_j), we require that every node $k, 1 \leq k \leq n$, belongs to some set $I_i, 1 \leq i \leq m$, in the final computation. We will then say that every node has had influence on the computation. This implies that every node must transmit at least once for each maximum computation to be complete.

Further, we define $\mathcal{N}_j^{(k)}$, the k hop neighborhood of node j as follows. Let $\mathcal{N}$ denote the set of n nodes. $\mathcal{N}_j^{(0)} := \{j\}$, $\mathcal{N}_j^{(1)} := \{i \in \mathcal{N} : |\mathbf{X}_i - \mathbf{X}_j| \leq r(n)\}$, $\cdots$ and $\mathcal{N}_j^{(k)} := \{i \in \mathcal{N} : |\mathbf{X}_i - \mathbf{X}_l| \leq r(n), l \in \mathcal{N}_j^{(k-1)}\}$. We note that, from the beginning of the slot in which node j first transmits its value, it takes *at least* k hops until the computations in the set $\mathcal{N}_j^{(k)} - \mathcal{N}_j^{(k-1)}$ are influenced by the value of node j, i.e., in each slot the influence of node j can spread by at most one hop.

A *computation algorithm* defines the sequence of message passing transactions, between specified transmitter-receiver pairs, that leads to the function being computed and the results delivered to the operator station. A computation algorithm may have associated with it a subgraph of $G(\mathcal{S})$ (see Definition 2.1) such that only the links in this subgraph are activated. For example, in a Tree algorithm (see Section V) a tree subgraph of $G(\mathcal{S})$ is defined and only the links in this tree are activated, progressing from the leaves up to the root.

Scheduling Assumptions: A synchronised *time slotted* system is assumed, with a packet transmission between any pair of nodes requiring one slot. For the purpose of obtaining our scaling results, we assume perfect scheduling of transmissions, i.e., in every slot certain links are scheduled and these transmissions are guaranteed to succeed. The perfect scheduler has a set of *maximal activation sets*, i.e., a set of transmitter-receiver pairs which can communicate simultaneously without violating the interference constraints. The activation sets that are scheduled are maximal in the sense that addition of any transmitter-receiver pair in such a set will violate the interference constraints. Also, the perfect scheduler is assumed to be optimal in the sense that given the node placements and the set of transmissions to be activated, it chooses a sequence of activation sets that schedules the transmissions in the minimum number of slots. Owing to these assumptions, our scaling laws should be viewed as bounds on what is practically achievable.

Computation and Scheduling Interaction: The computation progresses in *stages*, each stage requiring the transmission of messages from certain transmitters to designated receivers (including, possibly, multicasts to a set of receivers in the neighbourhood of each transmitter as in the Ripple algorithm (see Section V)). Given the transmissions to be scheduled at each stage, the scheduler provides a deterministic sequence of maximal activation sets that need to be activated in the successive slots in order to complete this stage of computation. Thus, a stage of a computation would be executed over several slots. After the completion of a stage in the computation, the computation algorithm defines the next set of transmissions to be scheduled.

For example, in the Tree algorithm (see Section V), each stage corresponds to the activation of links at one level in the tree. The scheduling algorithm then determines the number of slots required to compute this stage.

When a stage of computation involves one transmission from every node, we call such a stage a *round*. Note that since a computation algorithm requires each node to transmit its measured value at least once (see the discussion of the Distributed Computation Model above), a computation involves at least one round.

Energy Expenditure Model: We consider the following components of energy expenditure per packet transmission and reception. $E_{xmit-radio}$: the transmit energy radiated. Thus $E_{xmit-radio} = \alpha\, d^{\eta}$, where d is the distance between the transmitter and the receiver, η is the path loss exponent ($2 \leq \eta \leq 4$), and α is the energy corresponding to the received power level at the receiver required for successful reception in the presence of receiver noise (also sometimes called the receiver sensitivity). $E_{xmit-pkt}$: Energy required in the transmitter's electronics to transmit a packet. $E_{receive-pkt}$: Energy required in the receiver's electronics to receive a packet. $E_{proc-pkt}$: Energy required by the on-board computer to perform the computational task triggered by a received packet.

Performance Measures: For a given node placement $\mathcal{S}$, a compu-

tation algorithm along with the optimal centralized scheduler (see Computation and Scheduling Interaction above), will result in the maximum being computed in some number of slots. We denote this time required to complete the computation by $\Gamma(\mathcal{S})$. Thus, for a given computation algorithm, Γ is a random variable over Σ^n, which takes a specific value for every realization of $\mathcal{S}$. Also the node placement $\mathcal{S}$ and the computation algorithm (along with the centralised schedule) determine the number of transmissions and receptions by each node, and, thereby, the total energy spent. Let $E(\mathcal{S})$ be the total energy spent in the network while performing one computation, with E denoting the random variable over Σ^n, akin to Γ. *We seek "in probability" scaling laws, as n increases to ∞, for Γ and E for different computation algorithms.*

Remark: In [2], Giridhar and Kumar have addressed the problem of finding scaling laws for the rate of distributed computation (or data fusion). The functions they consider are *symmetric*[3]. Further, two subclasses of summetric functions, namely, *type-sensitive functions* and *type-threshold functions*[4] are considered. Scaling laws for upper and lower bounds on the computation rate are obtained in [2] for these subclasses of functions.

In our paper we consider only type-threshold functions. While [2] focuses on the the rate of computations, our focus in this paper is to obtain scaling laws for one time computation. Such a measure is of interest if a computation is triggered by a query. In addition, we also consider the energy expenditure for one-time computation as a performance measure and obtain scaling laws for it. Further, we also analyze three practical computation algorithms, namely, Tree, Multi-Hop and Ripple, with respect to the performance measures for one time computation. The reasons for studying such alternative algorithms are explained in Section V.

The overlap between our work and [2] is the optimal scaling law for the rate of computation. We view the rate of computation as an extension of the time for one-time computation. When the computations are pipelined, our message passing model treats each computation separately and does not maximally utilize the information available by virtue of the radio being a broadcast medium, and we do not exploit techniques such as block coding across measurements. Under these assumptions, it can be deduced from [2] and was also established independently in [5], that the maximum computation rate would be $\Theta\left(\frac{1}{\log n}\right)$. However, [2] further establishes that, for the class of type-threshold functions, we can take advantage of a technique like block coding to improve this bound to $\Theta\left(\frac{1}{\log \log n}\right)$. Thus, while the message passing model performs well for one-time computation, it is strictly suboptimal if the aim of the network is to perform pipelined computations.

However, it can be noted that the advantage of the block coding scheme of [2] over the message passing model can be exploited only when there are multiple computations pipelined in the network. When we consider only one time computation, block coding cannot be implemented and the scaling laws which can be obtained from the schemes in [2] will coincide with our scaling laws.

[3]A function is said to be symmetric if the function value is insensitive to the order in which data is processed or to the identity of node that measures the data.([2])

[4]Type-threshold functions were defined in an earlier footnote, and, informally, type-sensitive functions are those that require almost the entire data for computation, e.g., the mode of the measurements.

III. Background Results

The presentation of these results is interspersed with remarks about the intuition behind them. In writing these remarks we use the notation $\Theta(\cdot)$ loosely; it only means "of the order," the "in probability" qualification being implied.

Bounds on the number of hops in the shortest path: Consider a network realization $\mathcal{S}$ and consider all the pairs of points in the field $\mathcal{A}$ separated by a distance d; these points need not be locations of nodes. If a sender at one such point were to communicate with a receiver at the other point at a distance d, the packets will be transmitted along a multihop path using some intermediate nodes. The number of hops in the shortest path joining these points and using the intermediate nodes and links in $G(\mathcal{S})$ will be finite and will depend on the distance d. We define $\overline{H}(d)$ ($\underline{H}(d)$, resp.) as the supremum (infimum, resp.) over the number of hops in the shortest paths connecting all such pairs of points. Thus $\overline{H}(d)$ and $\underline{H}(d)$ are random variables over Σ^n defined for the distance d. We need probabilistic bounds on $\overline{H}(d)$ and $\underline{H}(d)$ as a function of d and $r(n)$. Evidently $1 \leq \underline{H}(d) \leq \overline{H}(d) \leq n+1$.

Lemma 3.1: For a *square* field of unit area, with $r(n) = \sqrt{\frac{K \log n}{n}}$, with $K > \frac{5}{\log \frac{4}{e}}$, the following holds for any $\epsilon > 0$.

1) $\lim_{n\to\infty} \mathcal{P}^n\left(G \text{ is connected }\right) = 1$
2) $\lim_{n\to\infty} \mathcal{P}^n\left(\frac{d}{r(n)} \leq \underline{H}(d) \leq \overline{H}(d) \leq (\sqrt{10}+\epsilon)\frac{d}{r(n)}\right) = 1$

Remark: This result has the obvious intuition. The transmission range of a node is $r(n)$. Hence, the number of hops in the shortest path between any two nodes separated by a distance d should be $\Theta\left(\frac{d}{r(n)}\right)$. We neglect edge effects in the proof.

Proof: The first part follows easily from the results in [6]. The proof of the second part is provided in the Appendix. □

Corollary 3.1: In a *circular* field of unit radius if n nodes are deployed and $r(n) = \sqrt{\frac{K\pi \log n}{n}}$, with $K > \frac{5}{\log \frac{4}{e}}$, the following holds for any $\epsilon > 0$.

1) $\lim_{n\to\infty} \mathcal{P}^n\left(G \text{ is connected }\right) = 1$
2) $\lim_{n\to\infty} \mathcal{P}^n\left(\frac{d}{r(n)} \leq \underline{H}(d) \leq \overline{H}(d) \leq (\sqrt{10}+\epsilon)\frac{d}{r(n)}\right) = 1$

Proof: For any convex field, only the node density, and not the shape of the field determines the relation between the transmission range and the number of hops. (See the proof of Lemma 3.1.) Thus, the result follows for the circular field also.

Remark: In this paper, we have chosen $K = 20 > \frac{5}{\log(\frac{4}{e})}$. Thus, from Corollary 3.1, we have $r(n) = \sqrt{\frac{20\pi \log n}{n}}$ for a circular field of unit radius, and from Lemma 3.1, we have $r(n) = \sqrt{\frac{20 \log n}{n}}$ for a square field of unit area. This form of $r(n)$, as established in [6] and [7], represents the fastest rate of decrease of $r(n)$ such that the probability of the random graph being connected approaches 1 as $n \to \infty$. Thus this scaling of $r(n)$ maximizes spatial reuse while retaining connectivity. We note that the choice of K does not influence the scaling laws.

Bounds on the number of simultaneous transmissions: For a given realisation $\mathcal{S}$, consider a round of the computation algorithm in which each node has to transmit once to a designated receiver node. For example, if the computation is performed over a tree in $G(\mathcal{S})$, then a round may involve *each* node sending a packet to its parent node. The centralised scheduler described above will schedule the transmissions by scheduling a sequence of activation sets. This sequence of activation sets will of course depend on a particular transmitter-receiver pairing. We are interested in obtaining bounds on the cardinalities of these activation sets. These are bounds on the number of simultaneous transmissions possible in the network. Consider all transmitter-receiver pairings in which each node appears as a transmitter exactly once. Over all such pairings, denote by $\underline{\gamma}(\mathcal{S})$ and $\overline{\gamma}(\mathcal{S})$, the minimum and maximum sizes of the activation sets scheduled by the scheduler. Thus, the quantities $\underline{\gamma}$ and $\overline{\gamma}$ also are random variables over Σ_n that take specific values for a given network realization.

Lemma 3.2: For a circular field of unit radius, with the protocol model of interference, and with perfect scheduling of transmissions, if the transmission range $r(n) = \sqrt{\frac{20\pi \log n}{n}}$, there exist positive constants a_1 and a_2 such that

$$\lim_{n\to\infty} \mathcal{P}^n \left(a_1 \frac{n}{\log n} \leq \underline{\gamma} \leq \overline{\gamma} \leq a_2 \frac{n}{\log n} \right) = 1$$

Remark:

1) We will provide the intuition for this result after Lemma 3.3.
2) The result holds for any field $\mathcal{A}$ with finite area, but we prove it only for the circular field of unit radius.

Proof: See the Appendix. □

Bounds on the time required for a round: The time required to complete a round of a computation will depend on the particular transmitter-receiver pairing. Let $\overline{T}(\mathcal{S})$ ($\underline{T}(\mathcal{S})$, resp.) denote the maximum (minimum, resp.) of the time required to complete a round, with maximum (minimum, resp.) being taken over all possible transmitter-receiver pairings. We note that for a given network realization $\mathcal{S}$ and optimal link scheduling algorithm, the bounds $\overline{T}(\mathcal{S})$ and $\underline{T}(\mathcal{S})$ are well defined. Thus, $\overline{T}$ and $\underline{T}$ are random variables over Σ^n.

Lemma 3.3: For a circular field of unit radius, with the protocol model of interference, with perfect scheduling of transmissions, and with the transmission range $r(n) = \sqrt{\frac{20\pi \log n}{n}}$, the bounds on the time required to schedule transmission of all the nodes $\overline{T}$ and $\underline{T}$ satisfy the relation

$$\lim_{n\to\infty} \mathcal{P}^n \left(5\pi\Delta^2 \log n \leq \underline{T} \leq \overline{T} \leq 80\pi(1+\mu)(2+\Delta)^2 \log n \right) = 1$$

Proof: See the Appendix. □

Remark: The above two results can be understood as follows. It has been shown in [7] that for simultaneous reception, the receivers should be at least $\frac{\Delta r(n)}{2}$ distance apart. Thus, the number of simultaneous transmissions is upper bounded by the number of disjoint disks of radius $\frac{\Delta r(n)}{2}$. This bound is tight only for a specific realisation $\mathcal{S}$ and specific transmitter-receiver pairing. It is easy to see that if the transmitters or the receivers are $(2+\Delta)r(n)$ apart, then any transmitter-receiver pairing is possible. Thus, very loosely, we can say that the number disjoint disks of radius $\frac{(2+\Delta)r(n)}{2}$ will lower bound the number of simultaneous transmissions. These disks have an area of $\Theta(\pi r^2(n))$. Hence the number of simultaneous transmissions should be $\Theta\left(\frac{1}{r^2(n)}\right) = \Theta\left(\frac{n}{\log n}\right)$. This implies that the number of slots required to schedule the transmissions of all the nodes once is $\Theta(\log n)$.

Farthest nodes: We will need the following observation about the sequence of random node locations characterised by the sequence of probability spaces Σ^n, $n \geq 1$.

Lemma 3.4: Consider a square field of unit area. For a given $\epsilon > 0$, the probability that the farthest node from the center of the field lies at a distance greater than $\left(\frac{1}{\sqrt{2}} - \epsilon\right)$ goes to 1 as $n \to \infty$.

IV. Optimal Orders for Performance Measures

In this section, we obtain the optimal order results for performance measures such as computation time, energy expenditure and the rate (throughput) of maximum calculation. Initially we obtain the results for a square of unit area and obtain a bound on the computation time. We then extend this to a circular field. The transmission range $r(n)$ in each case is as provided by Lemma 3.1 and Corollary 3.1.

For computation time and energy expenditure, we shall first obtain absolute lower bounds in order sense (i.e., we establish $\Omega(\cdot)$ relations). These bounds are absolute in the sense that no algorithm can do better than these bounds. We then construct centralized algorithms that achieve the *same* order as that of the lower bounds (but with a different leading constant). This gives an upper bound on computation time and energy expenditure for an optimal algorithm (i.e., $O(\cdot)$ relation). Thus we obtain an exact order (i.e., $\Theta(\cdot)$ relation) for an optimal algorithm.

Optimal order for computation time:

Theorem 4.1: If n nodes are uniformly distributed in a square field of unit area, then there exist positive constants u_1 and u_2 such that the number of slots required for an optimal algorithm to calculate the maximum measured value under the assumption of perfect scheduling obeys the following relation

$$\lim_{n\to\infty} \mathcal{P}^n \left(u_1 \sqrt{\frac{n}{\log n}} \leq \Gamma \leq u_2 \sqrt{\frac{n}{\log n}} \right) = 1$$

Proof: Let d_{max} denote the distance between the operator station and the farthest node in the network. Consider the event

$$\begin{aligned} & \left\{ \Gamma \geq \left(\frac{1}{\sqrt{2}} - \epsilon\right) \frac{1}{r(n)} \right\} \\ \supseteq \quad & \left\{ \Gamma \geq \underline{H}(d_{max}) \geq \left(\frac{1}{\sqrt{2}} - \epsilon\right) \frac{1}{r(n)} \right\} \\ \supseteq \quad & \{\Gamma \geq \underline{H}(d_{max})\} \cap \left\{ \underline{H}(d_{max}) \geq \frac{d_{max}}{r(n)} \right\} \cap \left\{ d_{max} \geq \frac{1}{\sqrt{2}} - \epsilon \right\} \end{aligned}$$

Consider the last expression above. We recall that we require that the computation must have the influence of all nodes in it and it takes at least $\underline{H}(d_{max})$ slots until the farthest node influences the computation at the operator station. Hence, and from Lemma 3.1, the first and second events have probability 1. From Lemma 3.4, the probability of the third event goes to 1 as $n \to \infty$. Combining these, for $\epsilon > 0$

$$\lim_{n\to\infty} \mathcal{P}^n \left(\Gamma \geq \left(\frac{1}{\sqrt{2}} - \epsilon\right) \frac{1}{r(n)} \right) = 1 \tag{1}$$

The upper bound on the computation time can be obtained by giving an actual computation algorithm. The details of the

algorithm are provided in the Appendix. Here we consider a centralized algorithm and obtain its computation time. The computation time of an optimal algorithm will be less than this time. □

The above result can be very easily extended to the circular field. We state the result as a corollary.

Corollary 4.1: If n nodes are uniformly distributed in a circular field of unit radius, then there exist positive constants v_1 and v_2 such that the number of slots required for an optimal algorithm to calculate the maximum measured value under the assumption of perfect scheduling obeys the following relation

$$\lim_{n\to\infty} \mathcal{P}^n \left(v_1\sqrt{\frac{n}{\log n}} \leq \Gamma \leq v_2\sqrt{\frac{n}{\log n}} \right) = 1$$

Optimal Order for Energy Expenditure:

We will now consider the optimal order for the energy expenditure. The result is stated below as a theorem. From the proof, it is clear that the energy expenditure does not depend on the shape of the field.

Theorem 4.2: If n nodes are uniformly distributed in a field, then the total energy expenditure in the network by an optimal algorithm to calculate the maximum measured value, under the assumption of perfect scheduling is $\Theta(n)$.

Proof: In any algorithm, every node has to transmit at least once and at least to one of its one hop neighbour. Hence, we get a lower bound as

$$E \geq n(E_{xmit-radio} + E_{xmit-pkt} + E_{receive-pkt})$$

We shall see that the Tree algorithm (see Section V) has energy expenditure

$$E_{Tree} = n(E_{xmit-radio}+E_{xmit-pkt}+E_{receive-pkt}+E_{proc-pkt})$$

An optimal algorithm will have energy expenditure at most equal to the Tree algorithm. Hence,

$$E \leq n(E_{xmit-radio} + E_{xmit-pkt} + E_{receive-pkt} + E_{proc-pkt})$$

Hence, $E = \Theta(n)$ and this is a deterministic scaling. □

Optimal Order for the Achievable Pipelined Throughput: Sometimes the network is required to perform the computations continuously. This can be viewed as a complex queueing system in which a batch of measurements arrives at sampling instants in the sampling buffers of the nodes with the arrival rate of the batches being equal to the sampling rate. The batch leaves the system when the corresponding maximum computation is over. It is of interest to obtain the saturation throughput of this system, which will dictate the permissible sampling rate of the sensors. That is, our interest is to characterize the interdeparture time of the batches of the measurements when the nodes are infinitely backlogged with measurements. We denote this inter-departure time (also called as pipelined computation time) by $\Gamma_{pipeline}(\mathcal{S})$. The sampling interval between the two measurements should be at least $\Gamma_{pipeline}(\mathcal{S})$. Thus, the reciprocal of the inter-departure time will give us the rate of computations, which we call pipelined throughput.

We view this as an extension of one time function computation where the computations are pipelined in the network. Given the realisation $\mathcal{S}$, the computation algorithm along with the scheduler comes up with a determistic sequence of transmissions which is periodic and which will complete the computations periodically with minimum inter-departure time. Thus, $\Gamma_{pipeline}$ is a random variable over Σ^n which takes value $\Gamma_{pipeline}(\mathcal{S})$ for a given $\mathcal{S}$.

We state the result for a square field of unit area (with the transmission range as per Lemma 3.1). The result for a circular field follows similarly.

Theorem 4.3: If n nodes are uniformly distributed in a square field of unit area, then there exist positive constants w_1 and w_2 such that the following relation bounds the pipelined computation time for an optimal algorithm performing pipelined maximum computations, under the assumption of perfect scheduling

$$\lim_{n\to\infty} \mathcal{P}^n \left(w_1 \log n \leq \Gamma_{pipeline} \leq w_2 \log n \right) = 1$$

Remark: We skip the proof of the above theorem as the result can be easily deduced from Theorem 2 in [2]. It has also been independently proved in [5]. Recall the observation that to complete a computation each node has to transmit at least once. From Lemma 3.3, the time required to schedule a round in a square field is $\Omega(\log n)$. The scheme used in obtaining the upper bound on the computation time in Theorem 4.1 can be used to show that the pipelined computation time is $O(\log n)$, thus proving the result. Let us call the in-network time of a computation as the time between the first transmission from a batch of measurements (anywhere in the network) and the time when the batch departs the network after computation is over. It has been shown in [5] that the *in-network* time of a batch of measurements in this algorithm also is of the order $\Theta\left(\frac{1}{r(n)}\right)$, i.e., $\Theta\left(\sqrt{\frac{n}{\log n}}\right)$, the same as the time for one time computation. A very similar result can also be established for a circular field.

V. Performance of Specific Protocols with Perfect Scheduling

In Section IV we provided scaling laws for the performance of an optimal algorithm. We analyze the performance of some specific computation algorithms in this section. We will provide scaling results and the intuition behind them. In this section, we consider a circular field with unit radius. The detailed proofs are omitted, and can be found in [5]. The motivation behind choosing these algorithms and their usefulness are mentioned in various remarks.

Let $D_i, 1 \leq i \leq n$, denote the distance of node i from the centre of $\mathcal{A}$ (i.e., from the operator station). Evidently $D_i, 1 \leq i \leq n$ is a sequence of i.i.d. random variables on Σ^n with a common distribution. Let $g_D(s)$ denote the density of this distribution. For a circular field of unit radius, this is easily seen to be $g_D(s) = 2s, 0 \leq s \leq 1$ (To see this, note that the CDF is $G_D(s) = \frac{\pi s^2}{\pi.1^2}$.)

Tree Algorithm: The communication topology is a tree with the operator station as the root. We call the nodes at the m^{th} level in the tree as parent nodes of the nodes at the $(m+1)^{th}$ level; and the nodes at the $(m+1)^{th}$ level as the children nodes of those at level m, with the root being the node at level 1. Here the children of a sensor are amongst its one hop neighbours. Each sensor gets values from its children, compares them with its own value and forwards only the maximum value to its parent. So, for each maximum computation, each sensor transmits only once. The slowest computation will be over a tree that is a string. For

a faster computation, we need a shallow tree. Hence, we take all the neighbours of the operator station as its children and build a breadth-first tree (See [3] for the properties of breadth-first trees).

We also note that nodes with different parents are not assured to have simultaneous transmissions. Simultaneous transmissions occur only if the nodes have different parents and the interference constraints are met.

The following result provides the asymptotic order for the computation time and energy expenditure for maximum computation over a breadth-first tree. The number of hops required for the farthest node to reach the centre is the depth of the tree. From Corollary 3.1, we know upper and lower bounds on the number of hops, i.e., the depth of the tree. We analyse Γ_{Tree} in both the cases, one for the lower bound on depth and the other for the upper bound on depth, which combined together provide the following result. A detailed proof is provided in [5].

Proposition 5.1: For the Tree algorithm, if n sensors are distributed uniformly over a circular field of unit radius, then there exist positive constants a_1, b_1 and b_2 such that the energy expenditure E and the computation time Γ required to compute the maximum of the measured values satisfy the relations.

$$E_{Tree} = \Theta(n). \quad \text{Moreover, } E_{Tree} = a_1 n \quad \text{as} \quad n \to \infty$$
$$\lim_{n\to\infty} \mathcal{P}^n \left(b_1\sqrt{n\log n} \le \Gamma_{Tree} \le b_2\sqrt{n\log n} \right) = 1$$

Remark:

1) **Intuition for the scaling law**: In the Tree algorithm every node transmits only once. There are n transmissions, n receptions and $n-1$ comparisons. Hence the energy expenditure is $\Theta(n)$. For computation time we know that the number of hops between the farthest node and the operator station is of the order $\frac{1}{r(n)}$. It can be shown ([5]) that the time required to schedule the nodes at a level is of the order $\log n$. Hence the computation time is of the order $\frac{1}{r(n)}\log n$, i.e. $\sqrt{n\log n}$. This is because the nodes at a level cannot be scheduled before all the descendant nodes of this level are scheduled.
2) **Discussion on the algorithm**: The Tree algorithm minimizes the number of transmissions required and the energy expended in computation by computing the functions in a distributed manner in the network. Clearly, it is best suited for sensor networks in terms of energy efficiency. The computation time depends largely on the structure of the tree and the Tree algorithm builds a breadth-first tree in the network.
   This algorithm requires the nodes to self organise into a tree and also to involve themselves in computations. Such self organizations may not always be desirable as tree maintenance (considering the practical fact of node failure) requires algorithms which will themselves require energy.

Multi-Hop Transmission: In this computation algorithm the value at each node is transported via multihop transmissions to the operator station. No computations are performed at the intermediate nodes. Each transmission follows a shortest path from a node to the operator station. Since the breadth-first tree gives the shortest path between the root and any node in the tree, this computation algorithm also involves a tree rooted at the operator station.

We recall that for circular field with unit radius, the transmission range $r(n) = \sqrt{\frac{20\pi\log n}{n}}$. Let $H(d)$ denote the random variable denoting the number of hops in the shortest path from a node at a distance d from the centre to the operator station. From Corollary 3.1, if a node is at a distance s from the center of the field, then for $\epsilon > 0$,

$$\lim_{n\to\infty} \mathcal{P}^n \left(\frac{s}{r(n)} \le \underline{H}(s) \le \overline{H}(s) \le (\sqrt{10}+\epsilon)\frac{s}{r(n)} \right) = 1$$

where $\underline{H}(s)$ and $\overline{H}(s)$ are lower and upper bounds the number of hops in the shortest path from that node to the operator station. We define

$$E_{hop} := E_{xmit-pkt} + E_{xmit-radio} + E_{receive-pkt}$$

where, $E_{xmit-radio} = \alpha \left(\sqrt{\frac{20\pi\log n}{n}} \right)^{\eta}$

The average energy spent in the node's transmission to the operator station is calculated as follows

$$\mathsf{E}(E) = \mathsf{E}(H(D).E_{hop}) \quad = \quad E_{hop} \int_0^1 \mathsf{E}(H(s))\, 2s ds \quad (2)$$

where $\mathsf{E}(\cdot)$ denotes expectation of a random variable. From the definition of $\overline{H}(s)$ and $\underline{H}(s)$, it follows that

$$\mathsf{E}(\underline{H}(s)) \le \mathsf{E}(H(s)) \le \mathsf{E}\big(\overline{H}(s)\big)$$

Consider

$$\begin{aligned} \mathsf{E}(\underline{H}(s)) &\ge \mathsf{E}\Big(\underline{H}(s).I_{\{\underline{H}(s)\ge\frac{s}{r(n)}\}}\Big) \\ &\ge \frac{s}{r(n)}.\mathcal{P}^n\left(\underline{H}(s) \ge \frac{s}{r(n)}\right) \end{aligned}$$

where $I_{\{.\}}$ is and indicator function. Now consider

$$\begin{aligned} \mathsf{E}\big(\overline{H}(s)\big) &= \mathsf{E}\left(\overline{H}(s).I_{\{\overline{H}(s)\le\frac{(\sqrt{10}+\epsilon)s}{r(n)}\}}\right) + \mathsf{E}\left(\overline{H}(s).I_{\{\overline{H}(s)>\frac{(\sqrt{10}+\epsilon)s}{r(n)}\}}\right) \\ &\le \frac{(\sqrt{10}+\epsilon)s}{r(n)}\mathcal{P}^n\left(\overline{H}(s) \le \frac{(\sqrt{10}+\epsilon)s}{r(n)}\right) \\ &\quad + n\mathcal{P}^n\left(\overline{H}(s) > \frac{(\sqrt{10}+\epsilon)s}{r(n)}\right) \end{aligned}$$

The above equation uses the fact that the maximum number of hops in shortest path between any node and the operator station cannot be more than n, the number of nodes. It can be shown that $\mathcal{P}^n\left(\overline{H}(s) > \frac{(\sqrt{10}+\epsilon)s}{r(n)}\right)$ is at most $O\left(\frac{1}{n}\right)$. Hence

$$\begin{aligned} \frac{s}{r(n)}\mathcal{P}^n\left(\underline{H}(s) \ge \frac{s}{r(n)}\right) &\le \mathsf{E}(H(s)) \\ &\le \frac{(\sqrt{10}+\epsilon)s}{r(n)}\mathcal{P}^n\left(\overline{H}(s) \le \frac{(\sqrt{10}+\epsilon)s}{r(n)}\right) + x_n \end{aligned}$$

where $x_n = O(1)$. Hence as $n \to \infty$, $\mathsf{E}(H(s)) = \Theta\left(\frac{s}{r(n)}\right)$. Thus substituting in Equation 2 and simplifying, we get $\mathsf{E}(E) = \Theta\left(\frac{1}{r(n)}\right)$. This gives the total energy, as $n \to \infty$, as

$$\mathsf{E}(E_{Multi-Hop}) = n\mathsf{E}(E) = \Theta\left(\frac{n}{r(n)}\right) \quad = \quad \Theta\left(n\sqrt{\frac{n}{\log n}}\right)$$

We now obtain bounds on the computation time $\Gamma_{Multi-Hop}$. Each sensor sends its value to the operator station via the shortest path. These paths can be found by constructing a breadth first tree

with operator station as root. Thus, the transmissions take place on the same tree as in the Tree algorithm but with the difference that now the values do not merge as they travel along the tree. The computation is complete when all the n measurements are received at the operator station. Since, the operator station can receive at most one packet in a slot, it will take at least n slots to complete the computation. Thus, we get the obvious lower bound on the computation time as

$$\Gamma_{Multi-Hop} \geq n \quad \text{for all } n$$

We now obtain an upper bound. The computation algorithm progresses in stages. In each stage, all nodes that have packets to send to the operator station transmit one packet to their parents in the tree. Thus in the first stage, all the nodes transmit. Let the number of neighbours of the operator station be n_0. (n_0 is a random variable defined over Σ^n that takes specific value for each network realisation.) Then in the first round the operator station receives n_0 new values; these are the values of its neighbours. In the next stage, the leaf nodes have no packets, but the next level of nodes will have one or more packets. During this stage again, the operator station receives n_0 new values. Then the number of stages are $\frac{n}{n_0}$ and the number of slots required for each stage is bounded by the number of slots required for a full round which is provided by Lemma 3.3. Note that all the nodes do not transmit in all the stages.

The number of neighbours of the operator station is of the order $\log n$ ([8]). A simple expression has been obtained in [5] that

$$\lim_{n\to\infty} \mathcal{P}^n(10\pi \log n < n_0 < 30\pi \log n) = 1 \tag{3}$$

Thus using Lemma 3.3 and Equation 3, we get bounds on the computation time as

$$\lim_{n\to\infty} \mathcal{P}^n(\frac{\Delta^2}{6} n < \Gamma_{Multi-Hop} < 8(1+\mu)(2+\Delta)^2 n) = 1$$

We summarize the above results as follows.

Proposition 5.2: For multi-hop transmission in a circular field with unit radius, as $n \to \infty$

$$\mathsf{E}(E_{Multi-Hop}) \quad = \quad \Theta\left(n\sqrt{\frac{n}{\log n}}\right)$$

and there exists a positive constant d_1 such that the computation time $\Gamma_{Multi-Hop}$ required to compute the maximum satisfies the relation,

$$\lim_{n\to\infty} \mathcal{P}^n\left(n \leq \Gamma_{Multi-Hop} \leq d_1 n\right) = 1$$

Remark:

1) **Intuition for the scaling law**: We know that the time required to schedule the transmissions in an area is of the order $\frac{n}{\frac{1}{r^2(n)}}$, i.e., $\log n$. After each round, the operator station receives one value from each of its one hop neighbours, the number of which is of the order $\frac{n}{\pi} \times \pi r^2(n) = \log n$, i.e., node density $\times$ the area covered by the transmission range. Hence the number of rounds required for receiving all the n values is of the order $\frac{n}{\log n}$. Multiplying by the time required for each round, we find that the computation time is of the order n. For, energy expenditure, we notice that each node's packet requires on an average $\Theta\left(\frac{1}{r(n)}\right)$ transmissions, one at each hop between the node and the operator station. Each transmission requires fixed energy. Hence, the energy expenditure is of the order $\frac{n}{r(n)}$, i.e., $n\sqrt{\frac{n}{\log n}}$.
2) **Discussion on the algorithm**: The Multi-Hop algorithm models traditional sensor networks where the nodes only collect the data and forward it to the central operator station where all the processing is done. Such a procedure requires nodes to form a topology, e.g., a tree rooted at the operator station, and to act as routers. This self-organization is not always desirable as maintenance of the tree requires energy. However, the advantage is that there is absolutely no restriction on the functions being computed, multiple functions can be computed for each set of sample values, and the functions to be computed can be easily changed.

Ripple Algorithm: In this algorithm, sensors keep on exchanging their current estimates of the maximum values and eventually all the sensors know the maximum value. The transmissions take place in stages that are rounds as defined in Section II. In a round, every node broadcasts its current estimate of the maximum value to from its neighbours; receives their estimates and then updates its own estimate of the maximum value at the end of the round. We note that the influence of the values of all the sensors propagates one hop distance in every round, like a Ripple. Hence, once the influence of the value of the farthest node reaches the centre, the computation is over. We need not continue until all the nodes know the actual maximum.

We note that what we call the Ripple algorithm is related to Gossip algorithms (see, for example, [10]) and Consensus algorithms (see, for example, [11]). In these algorithms, nodes exchange partial computations with their neighbours (synchronously or asynchronously) in order to compute some function of the values at nodes, so that eventually the function value is known to all the nodes. Thus with reference to the maximum computation problem in our paper we can say that Ripple achieves "max-consensus." The analyses of Gossip and Consensus algorithms in [10] and [11], however, do not consider the transmission scheduling aspect which is a key issue that we are concerned with in our work.

To analyse the performance of the Ripple algorithm, consider a circular field with unit radius. For an $\epsilon > 0$, the probability that the farthest node has a distance of at least $(1-\epsilon)$ unit from the centre approaches 1 as $n \to \infty$ (extension of Lemma 3.4). In Lemma 3.3, we have obtained bounds on the number of slots required to schedule a round. Since the influence of any node's value propagates by one hop in a round, the number of rounds required to complete the computation is calculated using Corollary 3.1 (with $\delta > 0$),

$$\lim_{n\to\infty} \mathcal{P}^n\left(\frac{(1-\epsilon)}{\sqrt{20\pi}}\sqrt{\frac{n}{\log n}} \leq \text{number of rounds} \leq \frac{(\sqrt{10}+\delta)}{\sqrt{20\pi}}\sqrt{\frac{n}{\log n}}\right) = 1 \tag{4}$$

This combined with Lemma 3.3 gives bounds on the computation time

$$\lim_{n\to\infty} \mathcal{P}^n\left((1-\epsilon)\frac{\sqrt{5\pi}\Delta^2}{2}\sqrt{n\log n} \leq \Gamma_{Ripple}\right.$$
$$\left. \leq (\sqrt{10}+\delta)4\sqrt{20\pi}(1+\mu)(2+\Delta)^2\sqrt{n\log n}\right) = 1$$

To calculate the energy expenditure, we need to know bounds

on the number of neighbours. For this we use a result from [8] to bound the number of neighbours.

In each round, every node broadcasts its current maximum value and receives its neighbours' values. It calculates the new maximum value and broadcasts it in the next round, i.e., only after it has got values from all neighbours. So, in a round each node has one transmission and receives and compares its neighbours' values to compute the current maximum. To find the maximum of m values, we need $(m-1)$ computations. If N_i is the number of neighbours of node i, then node i compares (N_i+1) values (neighbours' values and its own value). Hence, node i does N_i computations in each round.

The energy spent by node i in a round,

$$E_i = E_{xmit-radio} + E_{xmit-pkt} + N_i.E_{receive-pkt} + N_i.E_{proc-pkt}$$

where, $E_{xmit-radio} = \alpha\left(\sqrt{\frac{20\pi\log n}{n}}\right)^{\eta}$. Let $E_{xmitter} := E_{xmit-radio} + E_{xmit-pkt}$ and $E_{receiver} := E_{receive-pkt} + E_{proc-pkt}$.

From the bound on the number of neighbours from [8] (see Theorem 8.1 in Appendix), we obtain the following result for E_i

$$\lim_{n\to\infty} \mathcal{P}^n\left(E_{xmitter} + (1-\mu)20\pi\log n(E_{receiver}) \leq E_i \leq E_{xmitter} + (1+\mu)20\pi\log n(E_{receiver})\right) = 1$$

where μ satisfies the condition given in Theorem 8.1. This equation combined with the Equation 4 gives the bounds on E_i for any i which in turn gives the bound on E_{Ripple})

$$\lim_{n\to\infty} \mathcal{P}^n\left(\frac{(1-\epsilon)}{\sqrt{20\pi}} n\sqrt{\frac{n}{\log n}}\left(E_{xmitter} + (1-\mu)20\pi\log n.(E_{receiver})\right) \leq E_{Ripple} \leq \frac{\sqrt{10}+\delta}{\sqrt{20\pi}} n\sqrt{\frac{n}{\log n}}\left(E_{xmitter} + (1+\mu)20\pi\log n.(E_{receiver})\right)\right) = 1$$

We summarize the above results as a proposition.

Proposition 5.3: There are n sensors distributed independently and uniformly over a circular field of unit radius and the Ripple algorithm is used to compute the maximum value of the sensor measurements. There exist positive constants k_1, k_2, l_1 and l_2 such that the energy expenditure E_{Ripple} and the computation time Γ_{Ripple} required to compute the maximum satisfy the relations.

$$\lim_{n\to\infty} \mathcal{P}^n\left(k_1 n\sqrt{n\log n} \leq E_{Ripple} \leq k_2 n\sqrt{n\log n}\right) = 1$$

$$\lim_{n\to\infty} \mathcal{P}^n\left(l_1\sqrt{n\log n} \leq \Gamma_{Ripple} \leq l_2\sqrt{n\log n}\right) = 1$$

Remark:

1) **Intuition for the scaling laws**: We know that the number of slots required to schedule the transmissions in an area is of the order $\frac{n}{\frac{1}{r^2(n)}}$, i.e., $\log n$. After each round, the influence of the nodes' values propagate an additional hop. The number of hops between the farthest node and the operator station is of the order $\frac{1}{r(n)}$. Hence the rounds required are of the order $\frac{1}{r(n)}$. This shows that the computation time is of the order $\frac{1}{r(n)}\log n$, i.e. $\sqrt{n\log n}$. For energy expenditure, we notice that for each node in each round there is one transmission and $\Theta(\log n)$ receptions. As $n \to \infty$, the energy spent in reception is dominant over that of radio transmissions. Hence, the energy expenditure of the network is of the order $n\ \log n\ \frac{1}{r(n)}$, i.e., $n\sqrt{n\log n}$.

TABLE I

ORDER EXPRESSIONS FOR ENERGY EXPENDITURE AND COMPUTATION TIME FOR VARIOUS PROTOCOLS

Algorithm	Energy Expenditure	Computation Time
Optimum Algorithms	$\Theta(n)$	$\Theta\left(\sqrt{\frac{n}{\log n}}\right)$
Multi-Hop Algorithm	$\Theta\left(n\sqrt{\frac{n}{\log\ n}}\right)$	$\Theta(n)$
Tree Algorithm	$\Theta(n)$	$\Theta(\sqrt{n\log n})$
Ripple Algorithm	$\Theta(n\sqrt{n\ \log\ n})$	$\Theta(\sqrt{n\ \log\ n})$

TABLE II

EXPRESSIONS FOR ENERGY EXPENDITURE AND COMPUTATION TIME OBTAINED FROM SIMULATIONS WITH $\Delta = 0.5$

Algorithm	Energy Expenditure	Computation Time
Multi-Hop Algorithm	$0.24\left(n\sqrt{\frac{n}{\log\ n}}\right)$	$1.64(n)$
Tree Algorithm	$0.75(n)$	$3.15(\sqrt{n\log n})$
Ripple Algorithm	$1.44(n\sqrt{n\ \log\ n})$	$9.6(\sqrt{n\ \log\ n})$

2) **Discussion on the algorithm** : The Ripple algorithm is highly inefficient in computation time and the energy expended. However, in order to execute the algorithm, no specific topology needs to be discovered and maintained. In some applications this may be a more desirable mode of operation (e.g., applications where the nodes fail frequently). Also, we note that the Ripple algorithm provides the value of the function to every node in the network, which may be useful in an application where the entity that needs to use the results of the computation moves through the network. There is a restriction on the class of the functions that can be computed, however; the result of the computation should be insensitive to the repetitions of the arguments; e.g., $\max$, $\min$, k^{th} largest value etc. This is because each value measured at a node influences the partial results of various sets of nodes, and also returns to influence the results of the node that measured the value originally.

VI. SIMULATION RESULTS

Table I summarizes the order expressions obtained in the Section IV and V for all the algorithms. However, the constants multiplying these expressions are not known. In this section we validate these order results from a simulation, and as a by-product also obtain estimates of the constants.

The simulations are conducted as follows.

1) For the Tree algorithm, we build a breadth first tree rooted at the operator station. To calculate the computation time, we schedule the nodes at the same levels by building activation sets. We use the protocol model for interference. The schedule is suboptimal as we ensure that the transmissions are successful irrespective of the location of the receivers, i.e., all the transmitters are at least $(2+\Delta)r(n)$ distance apart from any other transmitter. We start from the leaf level and go on scheduling the nodes up the tree. This gives the computation time. Since, each node has only one fixed energy transmission, the energy can be readily calculated.

2) For Multi-Hop transmissions, the breadth first tree used in the Tree algorithm gives the shortest path to the root for each node. We note that all nodes do not always have packets to transmit. Hence, while scheduling, we consider only those nodes which have packets to send. The transmissions are carried on until all the n values reach the operator station. This gives the computation time. Each transmission requires fixed energy. Hence the total number of transmissions give the total energy expenditure.
3) In the Ripple algorithm, we have rounds in which each node transmits once. We schedule all the nodes using activation sets as before. In a round, each node has one transmission and as many receptions as the number of its neighbours. This gives the round time and energy per round which are the same for all rounds. Since the number of rounds required is the number of hops required for the farthest node to reach the operator station, the depth of the tree is the number of rounds required. This gives the total computation time and energy.

The simulation plots are obtained by taking an average over ten realizations of the node locations. The parameters used for the simulations are as follows: $E_{xmit-pkt} = 0.25$, $E_{receive-pkt} = 0.5$, $E_{proc-pkt} = 0.00001$, $\Delta = 0.5$, $E_{xmit-radio} = 100r^3(n)$ and $r(n) = \sqrt{\frac{2\pi \log n}{n}}$. The energy values can be viewed as scaled versions of practical values.

Suppose $\Phi(n)$ is the value of some performance measure obtained from the simulation as described above and $v(n)$ is the scaling law we obtain (e.g., $v(n) = \sqrt{n \log n}$ for the computation time for the Tree algorithm), then we plot $\frac{\Phi(n)}{v(n)}$. The theoretical results suggest the existence of constants $\underline{a}$ and $\overline{a}$ and functions $l(n) = o(v(n))$ and $u(n) = o(v(n))$ such that

$$\lim_{n\to\infty} \mathcal{P}^n \left(\underline{a}v(n) + l(n) \leq G(n) \leq \overline{a}v(n) + u(n)\right) = 1$$
$$\lim_{n\to\infty} \mathcal{P}^n \left(\underline{a} + \frac{l(n)}{v(n)} \leq \frac{G(n)}{v(n)} \leq \overline{a} + \frac{u(n)}{v(n)}\right) = 1$$
$$\lim_{n\to\infty} \mathcal{P}^n \left(\underline{a} \leq \frac{G(n)}{v(n)} \leq \overline{a}\right) = 1$$

Thus for large n, the values $\frac{G(n)}{v(n)}$ should be confined between the two limits $\underline{a}$ and $\overline{a}$.

In Figures 1, 2 and 3 we plot the ratios of the observed computation times and the asymptotic orders for the three algorithms. The plots show that in each case the interval within which these ratios should lie appears to collapse to a constant, an estimate of which is shown by the flat lines. The computation time has converged in all the cases.

The energy curves in case of Tree and Multi-Hop algorithms have not converged because for any finite n the $E_{xmit-radio}$ is finite. Since $E_{xmit-radio}$ decays with n, we observe that the energy curves are decaying. The energy curve in case of Ripple algorithm has converged because unlike the other two algorithm, Ripple has broadcast transmissions. Hence the total energy spent in reception dominates over $E_{xmit-radio}$. Of course, these observations also help to corroborate our scaling results.

In Table II we display the order results along with the constant multipliers estimated from the simulation.

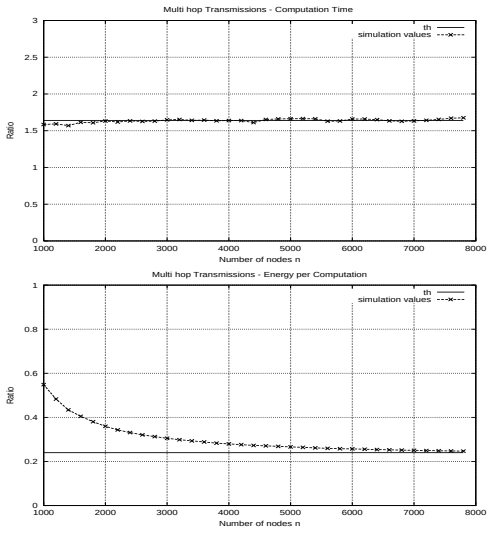

Fig. 1. Multihop transmission: (top panel) Ratio of observed computation time to the corresponding asymptotic order, and (bottom panel) the ratio of the observed energy expenditure per computation and the corresponding asymptotic order. The flat lines show the estimate of the constant multiplying the asymptotic order.

Figure 4 compares time and energy requirements of all the three algorithms. From these figures, it is clear that purely from the viewpoint of computation time and energy a tree is the preferable way to arrange the computations of the bounded information rate, recursively divisible functions. However, as discussed in Section V, there are several pros and cons to all the algorithms considered. Our results can guide a designer in weighing these against the performance measuresure we have studied.

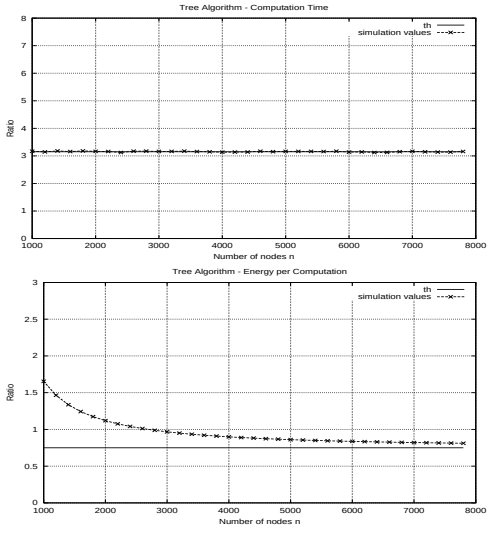

Fig. 2. Tree Algorithm: (top panel) Ratio of observed computation time to the corresponding asymptotic order, and (bottom panel) the ratio of the observed energy expenditure per computation and the corresponding asymptotic order. The flat lines show the estimate of the constant multiplying the asymptotic order.

VII. Conclusion

We have provided scaling laws for the computation time and energy consumption for the one shot distributed computation of

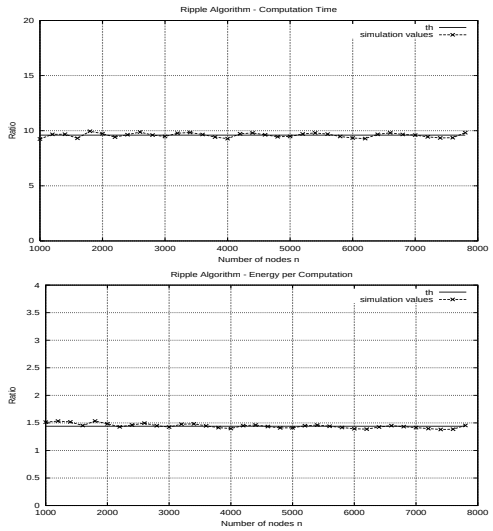

Fig. 3. Ripple Algorithm: (top panel) Ratio of observed computation time to the corresponding asymptotic order, and (bottom panel) the ratio of the observed energy expenditure per computation and the corresponding asymptotic order. The flat lines show the estimate of the constant multiplying the asymptotic order.

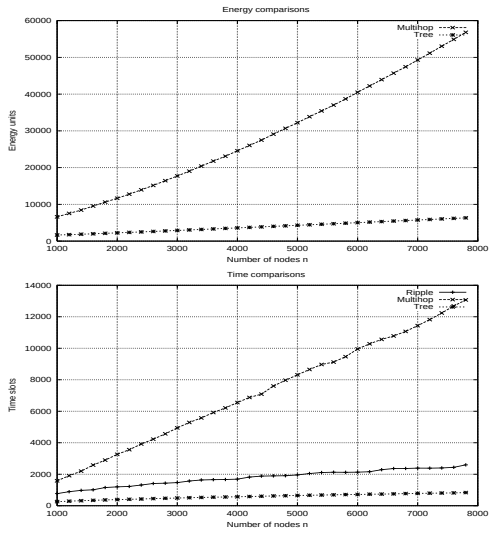

Fig. 4. Comparison of energy consumption (top panel) and computation time (bottom panel) per computation for the three algorithms. The energy required for the Ripple algorithm is not shown because of the very large values.

type-threshold functions. We have taken the $\max$ function as the representative member of this class that we use throughout in the presentation of our work. Our results are derived under the assumption of centralised optimal scheduling. We showed that the optimal (in probability) scaling law for the computation time is $\Theta\left(\sqrt{\frac{n}{\log n}}\right)$ and for energy expenditure is $\Theta(n)$ (Theorem 4.1 and 4.2). We also considered three practical algorithms and derive the scaling laws for the same measures. These are the Tree, Multi-Hop and Ripple algorithms. The scaling laws are derived in Propositions 5.1, 5.2 and 5.3, respectively, and are summarised in Table I. We also provide simulation results that corroborate the scaling laws and also provide the preconstants which are summarised in Table II. We also discussed some pros and cons of the various algorithms we have analysed.

We have assumed that the operator station is at the centre of the field, but even when the operator station is not at the centre, the distances between the farthest nodes and the operator station, and change by at most a constant factor, and hence do not change the scaling laws.

In the present work, we have assumed that the scheduler has global knowledge of the network topology. Hence, the results can be viewed as bounds on the performance when some distributed scheduler is implemented. The performance of distributed computing with a distributed medium access protocol (such as random access), which does not assume any global knowledge and algorithms for self-organisation of computing topologies, are topics of some of our ongoing work in this direction.

VIII. Appendix

Proof: (Lemma 3.1): Part(2):
Consider any two points A and B separated by distance d. Since the transmission range is $r(n)$, in one hop a packet can cover the distance of at most $r(n)$. Clearly, by the triangle inequality,

$$\mathcal{P}^n\left(\underline{H}(d) \quad \geq \frac{d}{r(n)}\right) = 1 \quad \text{for all} \quad n \tag{5}$$

To obtain an upper bound on the number of hops, we proceed as follows. We have a square field with unit area. We choose the transmission range $r(n) = \sqrt{\frac{K \log n}{n}}$, where $K > \frac{5}{\log \frac{4}{e}}$. We tessellate the field with a grid whose cells are of size $c(n) \times c(n)$ where $c(n) = \frac{r(n)}{\sqrt{5}} = \sqrt{\frac{K \log n}{5n}}$ as shown in Figure 5. Note that we have chosen $c(n)$ such that any two points in the neighbouring cells are within range of each other. (We call two cells as neighbouring cells if they share a common edge. Thus, an interior cell has four neighbouring cells.)

Now consider two points A and B separated by a distance d. Let θ be the angle made by the line segment AB with the horizontal axis. Consider the shortest path joining A and B. Clearly, if each cell is nonempty, the shortest path can go through at most $\lceil \frac{d \sin\theta}{c(n)} \rceil + \lceil \frac{d \cos\theta}{c(n)} \rceil$ cells. So if each cell is nonempty, there exists a path whose number of hops is bounded above by

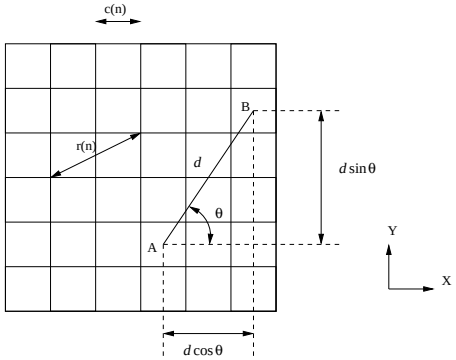

Fig. 5. The construction for obtaining the upper bound on the number of hops between any two nodes separated by distance d (e.g., the nodes A and B). Here, $c(n) = \sqrt{\frac{K \log n}{5n}}$ where $K > \frac{5}{\log \frac{4}{e}}$

$\lceil \frac{d\sin\theta}{c(n)} \rceil + \lceil \frac{d\cos\theta}{c(n)} \rceil \leq \frac{d\sin\theta}{c(n)} + \frac{d\cos\theta}{c(n)} + 2 \leq (\sqrt{10}+\epsilon)\frac{d}{r(n)}$ hops, where $\epsilon \geq \frac{2r(n)}{d}$. [5]

However, we need to ensure that each cell is nonempty with probability 1 as $n \to \infty$. Since we have chosen $c(n) = \sqrt{\frac{K \log n}{5n}}$ with $K > \frac{5}{\log \frac{4}{e}}$, it follows from a result by Xue and Kumar [8] (stated at the end of the Appendix as Theorem 8.1) that

$$\lim_{n\to\infty} \mathcal{P}^n \text{(Number of nodes in any cell} \geq (1-\mu)\log n) = 1$$

where $0 \leq \mu \leq 1$ satisfies certain conditions.

Thus,

$$\begin{aligned}
&\mathcal{P}^n\left(\bar{H}(d) \leq (\sqrt{10}+\epsilon)\frac{d}{r(n)}\right) \\
\geq\ & \mathcal{P}^n(\text{ There exists a path going through} (\sqrt{10}+\epsilon)\frac{d}{r(n)} \text{cells}) \\
\geq\ & \mathcal{P}^n(\text{No cell is empty}) \\
\geq\ & \mathcal{P}^n\text{(Number of nodes in any cell} \geq (1-\mu)\log n) \\
\to\ & 1 \quad \text{as} \quad n \to \infty
\end{aligned}$$

Thus,

$$\lim_{n\to\infty} \mathcal{P}^n\left(\bar{H}(d) \leq (\sqrt{10}+\epsilon)\frac{d}{r(n)}\right) = 1 \tag{6}$$

Equations 5 and 6 prove the lemma. □

Proof: (Lemma 3.2): In order to prove $\gamma = \Theta\left(\frac{n}{\log n}\right)$ asymptotically, we need two inequalities which we obtain in parts (a) and (b) as follows.

(a) Consider the two simultaneous transmissions $i \to j$ and $k \to l$. By the triangle inequality, it has been shown in [7] that the simultaneous transmissions $i \to j$ and $k \to l$ necessitate the condition $|X_j - X_l| \geq \Delta r(n)$. This means that the minimum distance between any two receivers receiving simultaneously must be at least $\Delta r(n)$, i.e., the disks of radius $\frac{\Delta}{2}r(n)$ centered around the receivers should be disjoint. [6].

The area of a disk of radius $\frac{\Delta}{2}r(n)$ is $\frac{\pi\Delta^2}{4}r^2(n)$. Hence, the trivial upper bound on the number of simultaneous transmissions in the field is the maximum number of the disjoint disks that can fit in $\mathcal{A}$.

$$\overline{\gamma}(\mathcal{S}) \leq \frac{\pi}{\frac{\pi\Delta^2}{4}r^2(n)} = \frac{1}{5\pi\Delta^2}\frac{n}{\log n} \tag{7}$$

The bound above actually is an unachievable upper bound since the actual number of the disks that can be accommodated in the field will be less than the above bound.

(b) To obtain the lower bound, we consider the construction shown in Figure 6. We inscribe a square field inside the circular field $\mathcal{A}$ and partition the square field into small square cells of sides $(2+\Delta)r(n)$ by a grid. We shall find a lower bound on the number of simultaneous transmissions possible only inside the square. Clearly, This is also a lower bound on $\underline{\gamma}(\mathcal{S})$. This approach avoids the cells which are at the boundary and are not completely within the field.

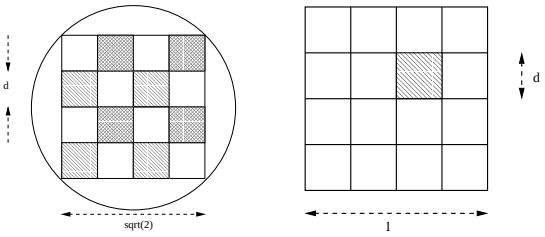

Fig. 6. (Left) Construction for lower bound on the number of simultaneous transmissions. $d = (2+\Delta)r(n)$. (Right) Tessellation into squares of side d to get the bounds on the number of neighbours of a node. Here $d = 2r(n) = \sqrt{\frac{K\log n}{n}}$ where $K = 80$.

From the Theorem 8.1 by Xue and Kumar [8], the probability that the number of nodes in any cell is between $20\pi(1-\mu)(2+\Delta)^2\log n$ and $20\pi(1+\mu)(2+\Delta)^2\log n$ goes to 1 as $n \to \infty$ [7]. Now we index the cells by their X and Y coordinates. The pair (i,j), where i and j are integers, denotes a cell. The origin can be chosen to be any cell. We mark the cells whose coordinates are of the form $(2l, 2k)$ (say, the hashed cells in Figure 6). We note that any node from one such marked cell is at least $(2+\Delta)r(n)$ distance away from any node in some other marked cell. Thus, if we schedule one node from each of the marked cells to transmit, these nodes can have simultaneous transmissions irrespective of the locations of their receivers. After scheduling these cells, we can schedule the transmissions of the nodes in the cells with coordinates of the form $(2l+1, 2k+1)$ (the filled cells in this case), $(2l, 2k+1)$ and $(2l+1, 2k)$ (unfilled cells) in three slots. Thus in four slots, a node from each cell is scheduled. The fact that the transmissions are successful irrespective of the location of the receivers gives a lower bound on the number of simultaneous transmissions possible.

The sides of the square are $\sqrt{2}$. Thus the total number of cells that can be accommodated in the square is $\frac{2}{(2+\Delta)^2 r^2(n)} = \frac{1}{10\pi(2+\Delta)^2}\frac{n}{\log n}$. In one slot, nodes from only $\frac{1}{4}^{th}$ of the cells can transmit. We also need to show that the probability of any cell being empty is zero as $n \to \infty$. This is also evident from Theorem 8.1. Thus

$$\lim_{n\to\infty} \mathcal{P}^n\left(\frac{1}{40\pi(2+\Delta)^2}\frac{n}{\log n} \leq \underline{\gamma}\right) = 1 \tag{8}$$

Thus, Equations 7 and 8 prove the lemma. □

Proof: (Lemma 3.3): In a round, each node transmits once to one of its neighbours. There are n transmissions in a round. In Lemma 3.2 we have obtained the bounds on the number of simultaneous transmissions in the network. We will use these bounds to get the bounds on the round time.

A lower bound can be obtained from Equation 7. We know that the number of simultaneous transmissions is bounded as $\overline{\gamma}(\mathcal{S}) \leq \frac{1}{5\pi\Delta^2}\frac{n}{\log n}$. Hence a lower bound on the round time $\underline{T}(\mathcal{S})$ for all $\mathcal{S}$ is

$$\underline{T}(\mathcal{S}) \geq \frac{n}{\overline{\gamma}(\mathcal{S})} = 5\pi\Delta^2 \log n \tag{9}$$

[5]Here we have used the fact that $\sin\theta + \cos\theta \leq \sqrt{2}$ for $0 \leq \theta \leq \frac{\pi}{2}$ and that $c(n) = \frac{r(n)}{\sqrt{5}}$. As $n \to \infty, r(n) << d$ and $\frac{2r(n)}{d} \to 0$.

[6]For this condition to suffice for the successful receptions at all the receivers, the transmitters and receivers must be located in a specific manner, which is the most compact arrangement of receivers possible.

[7]One can easily show that the condition on K in Theorem 8.1 is satisfied in this case.

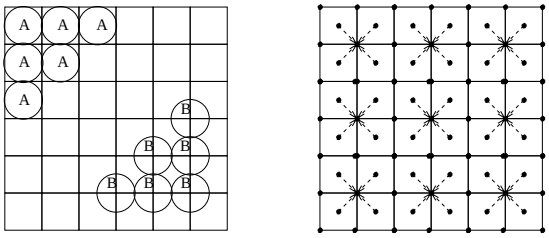

Fig. 7. The construction for an upper bound on the computation time.

To find an upper bound, we follow the construction in the proof of Lemma 3.2 and split the field $\mathcal{A}$ into squares with sides $(2+\Delta)r(n)$. We know that in four slots all the cells get one transmission scheduled, hence the time required to complete a round is determined by the maximum number of nodes a cell can have. From Theorem 8.1, we know that $20\pi(1+\mu)(2+\Delta)^2\log n$ bounds this number with high probability. Hence we get an upper bound on the round time.

$$\lim_{n\to\infty} \mathcal{P}^n(\overline{T} \leq 80\pi(1+\mu)(2+\Delta)^2\log n) = 1 \tag{10}$$

Equations 9 and 10 together prove the lemma. □

Proof: (Construction of a computation algorithm to achieve the upper bound on computation time given in Theorem 4.1):

For the simplicity of presentation, we assume that there is no interference outside the transmission range, i.e., $\Delta = 0$ in protocol model of interference. This assumption does not affect the order of the computation time. (See the remark below.) We divide the field into the square cells of sides $2r(n) = \sqrt{\frac{K\log n}{n}}$, where $K = 80$, as shown in Figure 6. The number of cells in the field $M_n = \frac{n}{K\log n}$. It follows from Theorem 8.1 that the number of nodes in a cell is bounded by $(1+\mu)K\log n$ w.p. 1 as $n \to \infty$.

Now, we draw circles as shown in the left part of Figure 7. As shown in Figure 7, we classify these circles as Type A and Type B circles. Type A circles have the centres of the cells as their centres and Type B circles have the corners of the cells as their centres. Type A and B circles together cover the whole area. Also,

$$\begin{aligned} & \mathcal{P}^n\,(\text{Number of nodes in any circle} \leq (1+\mu)K\log n) \\ \geq\ & \mathcal{P}^n(\text{Number of nodes in any cell} \leq (1+\mu)K\log n) \\ =\ & 1 \ \text{ as } \ n\to\infty \end{aligned}$$

Hence,

$$\lim_{n\to\infty} \mathcal{P}^n(\text{Number of nodes in any circle} \leq (1+\mu)K\log n) = 1$$

We choose the node which is nearest to the center of the circle as the cluster-head. Note that Type A and Type B circles overlap. Let all nodes in Type A circles form clusters, which we call type A clustures, and the nodes in Type B circles that do not lie in the Type A circles form type B clusters. Thus Type B clusters have a smaller number of nodes.

It can be shown similar to Lemma 3.4 that the probability that the cluster-head lies in the circle of radius ϵ at the center of the circle goes to 1 as $n \to \infty$ for all values of ϵ and hence when $\epsilon \to 0$.

All nodes within a circle can reach their cluster-heads in one hop. Since we have assumed zero interference outside the transmission range, non-overlapping circles can have simultaneous transmissions. This means all Type A clusters can have one transmission in each cluster simultaneously. After Type A cluster-heads have received the values from the cluster members (which will take at most $(1+\mu)K\log n$ slots), the same procedure can be repeated for Type B clusters. Hence, all the cluster-heads will get the values in their clusters in less than $2(1+\mu)K\log n$ slots and will compute the local partial results.

The Type A cluster-heads will now report their values to the assigned Type B cluster-heads as shown in the right side of Figure 7. Each Type A cluster-head has to send the value to a node which is at a distance $\sqrt{2}r(n)$ apart. Hence, the path will have $\leq (\sqrt{20}+2) = 2(\sqrt{5}+1)$ hops with high probability as $n \to \infty$ (Lemma 3.1). Thus, it will require at most $8(\sqrt{5}+1)$ slots. As the interference is assumed to be zero outside the distance $r(n)$ and the transmitting Type A cluster-heads assigned to different Type B cluster-heads are at least at a distance $2r(n)$, these transmissions can be scheduled simultaneously.

Now only Type B cluster-heads have the partial results. They are aligned in the straight lines. The Type B cluster-heads which are near the left and right edge of the line transmit their values towards the central part (of the line) horizontally. As the values reach other cluster-heads in the path, the new maximum value is propagated ahead. These transmissions can occur simultaneously as the paths are confined in the squares of side $c(n) = \frac{r(n)}{\sqrt{5}}$ (Proof of Lemma 3.1), and the minimum distance between any two squares is $2r(n) - \frac{2r(n)}{\sqrt{5}} > r(n)$. Hence there is no interference.

Since the transmission range is $r(n)$ and the values have to propagate a distance 1/2 units, the probability that the time required for the propagation $\leq \frac{\sqrt{10}+\epsilon}{2r(n)}$ slots approaches 1 as $n \to \infty$.

Once these values merge on the central vertical line, the same procedure can be used to get all the values at the center. Considering that the probability of occurrence of all the events approaches 1 as $n \to \infty$ we can say that

$$\lim_{n\to\infty} \mathcal{P}^n\left(\Gamma \leq 2(1+\mu)K\log n + 8(\sqrt{5}+1) + \frac{\sqrt{10}+\epsilon}{2r(n)} + \frac{\sqrt{10}+\epsilon}{2r(n)}\right) = 1$$

$$\lim_{n\to\infty} \mathcal{P}^n\left(\Gamma \leq 160(1+\mu)\log n + 8(\sqrt{5}+1) + (\sqrt{10}+\epsilon)\sqrt{\frac{n}{20\log\ n}}\right) = 1$$

From Equations 1 and 11,

$$\lim_{n\to\infty} \mathcal{P}^n\left(\left(\frac{1}{\sqrt{2}} - \epsilon_1\right)\sqrt{\frac{n}{20\log n}} \leq \Gamma \right.$$

$$\left. \leq 160(1+\mu)\log n + 8(\sqrt{5}+1) + (\sqrt{10}+\epsilon)\sqrt{\frac{n}{20\log\ n}}\right) = 1$$

□

Remark: The assumption of $\Delta = 0$ is to simplify the presentation, and does not change the order of the computation time. If $\Delta > 0$, then the transmissions in a Type A cluster will interfere with some constant number of other Type A clusters and this constant depends only on Δ. Thus, in this case, all the Type A clusters can be activated in a constant number of slots (which depends only on Δ), rather than in 1 slot as assumed in the proof. Thus, we obtain the same scaling order even if $\Delta > 0$.

Intuition about the $\log n$ term in the upper bound expression can be obtained as follows. Since we form clusters of nodes that are one hop neighbours of the cluster-heads, the number of cluster members is the number of nodes that lie in a circle of radius $r(n)$ drawn around the cluster-head. The node density is n. Hence, the number of cluster members would be $\Theta(\pi r^2(n)n) = \Theta(\log n)$. Since only one node in a cluster can transmit in a slot, $\Theta(\log n)$ slots are needed to complete the collection of values from a cluster. The clusterheads need to transmit the values to the operator station. Since the clusterheads are sparsely distributed, simultaneous transmissions are possible and value of the farthest clusterhead needs to travel unit distance which requires $\Theta(\frac{1}{r(n)})$ slots.

We have extensively used the following result established by Xue and Kumar [8] stated here for completeness. In this set up, the square field of unit area is split into small cells of size $\sqrt{\frac{K\log n}{n}} \times \sqrt{\frac{K\log n}{n}}$ by a grid as shown in Figure 6. The cells are indexed by $i \in \{1, \ldots, \frac{n}{K\log n}\}$.

Theorem 8.1 (Xue and Kumar [8]): Let $K > \frac{1}{\log\left(\frac{4}{e}\right)}$, and let $\mu^* \in (0,1)$ be the only root of the equation

$$-\mu^* + (1+\mu^*)\log(1+\mu^*) = 1/K$$

We tessellate the square field of unit area as mentioned above. There are n nodes deployed uniformly in the square field. Let N_i be the number of nodes in the i^{th} cell; and M_n be the total number of cells $\left(M_n = \frac{n}{K\log n}\right)$. Then the following holds for any $\mu > \mu^*$

$$\lim_{n\to\infty} \mathcal{P}^n\left(\max_{1\le i\le M_n} |N_i - K\log n| \le \mu K\log n\right) = 1$$

Remark: This implies that the number of nodes lying in any cell is uniformly bounded between $(1-\mu)K\log n$ and $(1+\mu)K\log n$ w.p. 1 as $n\to\infty$. We note that the expected number of nodes in a cell is $K\log n$; thus μ captures the range of variation from the mean. We note that this result also holds in our case of circular field of unit radius.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramanian, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

[2] Arvind Giridhar and P. R. Kumar. Data fusion over sensor networks: Computing and communicating functions of measurements. *IEEE Journal on Selected Areas in Communications*, pp. 755–764, vol. 23, no. 4, April 2005.

[3] T. Cormen, C. Leiserson, R. Rivest. Introduction to Algorithms. *Second Edition, Prentice-Hall of India Pvt. Ltd.*

[4] Aditya Karnik and Anurag Kumar. Distributed optimal self-organisation in a class of ad hoc sensor networks. In *IEEE Infocom*, 2004.

[5] Nilesh Khude. Distributed computation on wireless sensor networks. Master's thesis, ECE Department, Indian Institute of Science, Bangalore, June 2004.

[6] Piyush Gupta and P.R. Kumar. Critical Power for Asymptotic Connectivity in Wireless Networks. A Volume in Honour of W.H.Fleming in Stochastic Analysis, Control, Optimisation and Applications, 1998.

[7] Piyush Gupta and P.R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, IT46(2):388–404, March 2000.

[8] Feng Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, pp. 169–181, vol.10, no. 2, March 2004.

[9] Nilesh Khude, Anurag Kumar and Aditya Karnik. Time and Energy Complexity of Distributed Computation in Wireless Sensor Networks. In *IEEE Infocom*, 2005.

[10] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah. Gossip Algorithms: Design, Analysis and Applications. In *IEEE Infocom*, 2005.

[11] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, Convergence in Multiagent Coordination, Consensus, and Flocking. in *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*, 2005.