Towards a Theory of In-Network Computation in Wireless Sensor Networks

Arvind Giridhar and P.R. Kumar
Department of Electrical and Computer Engineering, and
Coordinated Science Laboratory,
University of Illinois, Urbana-Champaign.
Email: {giridhar,prkumar}@uiuc.edu

***Abstract*— Sensor networks are not just data networks with sensors being the sources of data. Rather, they are often developed and deployed for a specific application, and the entire network operation is accordingly geared towards satisfying this application. For overall system efficiency, it may be necessary for nodes to perform computations on data, as opposed to simply originating or forwarding data.**

Thus, the entire network can be viewed as performing an application specific distributed computation. The topic of this paper is to survey some lines of research which may be useful in developing a theory of in-network computation, that aims to elucidate how a wireless sensor network should efficiently perform such distributed computation.

We review several existing approaches to computation problems in network settings, with a particular emphasis on the communication aspect of computation. We begin by studying the basic two-party communication complexity model and how to optimally compute functions of distributed inputs in this setting. We proceed to larger multi-hop networks, and study how block-computation and function structure can be exploited to provide greater computational throughput. We then consider distributed computation problems in networks subject to noise. Finally, we review some randomized gossip based approaches to computing aggregate functions in networks.

These are diverse approaches spanning many different research communities, but together may find a role in the development of a more substantial theoretical foundation for sensor networks.

I. Introduction

The purpose of traditional data networks such as the Internet is to enable end-to-end information transfer. Information streams in such networks are carried across point-to-point links, with intermediate nodes simply forwarding data packets without modifying the payload of the packets. In contrast, the purpose of a sensor network is to provide to users access to the information of interest from data gathered by spatially distributed sensors. In most applications, users require only certain aggregate functions of this distributed data. Examples include the average temperature in a network of temperature sensors, or a particular trigger in the case of an alarm network, location of an event, etc.

This material is based upon work partially supported by AFOSR under Contract No. F49620-02-1-0217, NSF under Contract Nos. NSF ANI 02-21357 and CCR-0325716, USARO under Contract Nos. DAAD19-00-1-0466 and DAAD19-01010-465, DARPA/AFOSR under Contract No. F49620-02-1-0325, and DARPA under Contact Nos. N00014-0-1-1-0576.

Such aggregate functions could be computed under the end-to-end information flow paradigm by communicating all the relevant data to a central collector node. This however is a highly inefficient solution in large sensor networks which may be severely constrained in energy, memory or bandwidth, and where tight latency constraints may have to be met.

The alternate solution is for the computation to be performed *in-network*. Implementing such distributed computing functionality in sensor networks carries a whole host of networking issues, because nodes have to not only originate and forward data, but may also have to perform operations on data received from different sources at different times. Some of this functionality is part of applications which provide support for *aggregate queries* over sensor networks [1].

At a fundamental level, therefore, resides the question of how best to perform distributed computation over a network of nodes with wireless links. What is the optimal way to compute the average of a set of statistically correlated values stored by different nodes of a wireless network? How would such computations be performed in the presence of unreliability such as noise, packet drops and node failures? Such questions combine the complexities of multi-terminal information theory, distributed source coding, communication complexity and distributed computation.

The focus of this paper is to provide an account of some of the work that may be relevant as we move towards such a theory of distributed computation over networks. These include the following:

1) Computing in the two node network: We review two-party communication complexity theory, which addresses the question of how to compute in minimum time a function with input distributed between two separate nodes.
2) Block computation over multiple node networks: We will review recent work extending the basic communication complexity model in two aspects, by modeling more general multiple node topologies, including single hop collocated networks as well as random multi-hop networks, and by allowing for the possible efficiencies of block-computation in which the function of interest is required to be repeatedly computed for different sets of measurements. We will show that both can lead to

a significant performance improvement over one-shot function computation.

3) Distributed computation in the presence of noise: The basic communication complexity model does not allow for noisy communication channels. We will review work on computing functions in a noisy collocated network on simulating general interactive protocols for noiseless networks over noisy networks. We will also describe some classical information theoretic formulations which incorporate block computation along with correlated sensor measurements, and review what little progress has been made towards solving such a general formulation.
4) Randomized algorithms for computing functions: We then turn to some recent work on gossip based protocols to compute aggregate functions over peer-to-peer as well as sensor networks. These approaches involve designing algorithms for computing specific functions, which may not be optimal, but are designed to be simple, scalable and robust to node failures.

In most of the topics we consider, the emphasis is on optimizing the communication aspect of computation. Given the physical constraints of sensor networks, such as limited energy, bandwidth, and lack of infrastructure, communication is one bottleneck to performance. However, there are others, such as energy and latency, which we do not address here. Also, we assume that that computation at nodes is reliable, while communication is subject to errors.

II. Computing in the Two-Node Network

In many applications, users of a sensor network require a certain function of the sensors' raw measurements. We may formalize this by stating the problem as one of computing a function $f(x_1, x_2, \ldots, x_n)$ of the readings $x_1, x_2, \ldots, x_n$ taken at the n sensor nodes, and communicating this function to a collector node. In a typical scenario, there may be only a single collector node, but it is also possible that there are multiple collector nodes. In fact, in an extreme situation, the function may need to be communicated to all the nodes.

The question of how to compute a function depends on a number of network features, such as its spatial distribution and size, the communication constraints, and of course the structure of the function itself. We consider first the case of a network of two nodes.

The basic two party communication complexity model is constructed to study distributed computation by separate processors connected by constrained communication links. For a detailed exposition of the theory, the reader is referred to [2]; here we provide a brief introduction. Let A and B be two processors connected by a bidirectional one-bit-per-slot link. A knows a number x which lies in a finite set X, and B a number y belonging to finite set Y. Both wish to compute the value of a certain function $f(x, y)$, which takes values in a set Z.

In order to compute this function, they can exchange bits over the link, one at a time. We are interested in strategies that for any allowable pair x, y, will terminate when both A and B know the correct value $f(x, y)$. The communication complexity of the function $f(\cdot)$ is the minimum over all such strategies, of the maximum time over all input pairs, before the strategy terminates. To summarize, this problem is one of minimizing computation time given a throughput constrained link between processors, and an input split between processors.

A general deterministic protocol must function as follows: It must sequentially specify for each slot, as a deterministic function[1] of all the previously transmitted bits, which node must transmit. It must also specify the value of the bit to be transmitted by a node, as a deterministic function of its input value and all the previous transmissions (which it has either received or sent).

One can derive a number of elegant results characterizing the communication complexity of a function. We review some of the more basic of these below. Let us begin with the most naive protocol, which simply consists of A sending the value of x, followed by B sending back the value of the function $f(x, y)$. This gives a trivial upper bound on the communication complexity of the function $f(\cdot)$ of $\log|X| + \log|Z|$ slots.

It can also be shown that $\lceil \log|Range(f)| \rceil$ is a lower bound on communication complexity, where $Range(f)$ is the set of values taken by f on $X \times Y$. The proof is as follows. It is enough to prove that under any correct protocol, any two distinct values of $f(\cdot)$ must correspond to different sequences of transmitted bits, since there would then have to be $|Range(f)|$ distinct transmitted sequences. Suppose that this is not the case, and that for inputs (x_1, y_1) and (x_2, y_2) with $f(x_1, y_1) \neq f(x_2, y_2)$, the sequences of transmitted bits specified by the protocol are the same. It cannot be that both $f(x_1, y_1) = f(x_1, y_2)$ and $f(x_1, y_2) = f(x_2, y_2)$ are true, because that would imply $f(x_1, y_1) = f(x_2, y_2)$. Suppose without loss of generality that $f(x_1, y_1) \neq f(x_1, y_2)$. Then, the definition of a protocol will imply that it specifies the very same sequence of transmissions for the input pair (x_1, y_2) as it does for the input pairs (x_1, y_1) and (x_2, y_2). This leads to a contradiction, since processor A cannot distinguish between the pairs (x_1, y_1) and (x_1, y_2), due to the fact that its own input value as well as the sequence of transmissions are the same in each case. Therefore, processor A cannot "know" whether the function value is $f(x_1, y_1)$ or $f(x_1, y_2)$.

The informal definition of a protocol given above can be formalized into a convenient matrix representation. The computation problem is entirely specified by the $|X| \times |Y|$ matrix C, where $C_{ij} = f(i, j)$ (assuming without loss of generality that $X = \{1, 2, \ldots, |X|\}, Y = \{1, 2, \ldots, |Y|\}$). Any protocol necessarily specifies a sequence of successive partitions of the matrix C. The first transmission partitions either the rows or columns of C (rows if A is selected as the first to transmit, columns if B is), thereby obtaining two sub-matrices. The next transmission corresponds to a separate partitioning of each of the two previously obtained sub-matrices, again along the rows or columns. Thus, the k^{th} stage specifies partitions separately for each of the 2^{k-1} sub-matrices specified by the previous

[1] Randomized algorithms have also been considered in this setting.

A/B	1	2	3	4
1	0	0	0	1
2	0	0	0	1
3	0	0	0	0
4	0	1	1	1

Fig. 1. Example of a protocol

$k-1$ stages, producing 2^k sub-matrices. The protocol ends when all the sub-matrices are "monochrome," i.e., have all identical entries. The communication complexity is therefore the minimum number of rounds in which the matrix C can be thus partitioned into monochromatic sub-matrices.

An example of such a partitioning is shown in Figure 1. This corresponds to A transmitting first, B transmitting second. If the input pair lies in the first two rows, or the first column of rows 3 and 4, then the function value is determined after these two transmissions. Otherwise, A must transmit again, after which the protocol has terminates. Thus, this protocol requires 3 time-steps.

A simple linear algebraic argument can now be used to lower bound the number of partitions required. At each round, the maximum rank of the resulting sub-matrices is decreased by at most a factor of two. After the final stage all sub-matrices have rank 1. Therefore, the communication complexity must be lower bounded by $\lceil \log Rank(C) \rceil$.

Another simple bound can be derived by the so-called *fooling set* technique. Suppose we have m input pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, with the property that $f(x_1, y_1) = f(x_2, y_2) = \ldots = f(x_m, y_m)$, but that for any two pairs (x_i, y_i) and (x_j, y_j), either $f(x_i, y_j)$ or $f(x_j, y_i)$ differs from $f(x_i, y_i)$. Then, by an argument similar to the range bound argument described above, one can prove that under any correct protocol each of these m input pairs must correspond to a unique sequence of transmissions, and so $D_f(f) \geq \log m$.

The following are some examples of functions and their communication complexities. For proofs the reader is referred to [2].

1) A and B have $D = 2^n$ valued inputs x and y.
   - $f(x, y) = Eq(x, y)$, defined as equal to 1 if $x = y$, 0 if not. The communication complexity of $Eq(\cdot)$ is $\log D + 1$. A lower bound of $\log D$ follows directly from the rank bound, since the matrix C in this case is the $D \times D$ identity matrix. A slightly stronger version of the rank argument gives a lower bound of $n + 1$.
   - $f(x, y) = Gt(x, y)$, defined as equal to 0 if $x > y$ and 1 otherwise. The communication complexity is $\log D + 1$.
2) A and B hold subsets $x, y \subseteq \{1, 2, \ldots, n\}$.
   - $f(x, y) = Avg(x, y)$, the average value of the two subsets. The communication complexity is then $O(logn)$.
   - $f(x, y) = Disj(x, y)$, which is equal to 0 if the subsets are disjoint, and 1 otherwise. The communication complexity is $n + 1$.
   - $f(x, y) = Med(x, y)$, the median of x and y. The communication complexity is $O(\log n)$.

Communication complexity theory provides a useful stepping stone in terms of understanding protocols for computing functions with distributed inputs. However, important features of sensor networks are not captured in this model. One necessary extension would be obtaining bounds in networks of more than two nodes. There has been some work on multi-party communication complexity in networks. Tiwari [3] considered a formulation in which the input is split among two processors which are nodes in a larger network. In the next section, we will describe some recent work on communication complexity problems in different wireless network models [4].

Another limitation of the basic communication complexity setup is that due to the requirement of exact computation, possible correlations between the inputs are not exploited. Such correlations are particularly likely to exist in large spatially dense networks sampling physical characteristics (e.g. temperature) of a domain.

III. Block Computation of Functions in Multi-party Networks

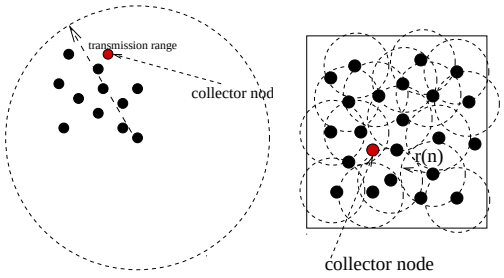
We now turn to a more general network setting, consisting of many nodes and a simplified model of wireless communication. Consider a network of n nodes located on a plane, along with a designated collector node to which the aggregate function of interest needs to be communicated.

Each node has a certain transmission range, and can transmit directly to any other node which is located within that range. The *connectivity graph* of the network is composed of the nodes along with edges modeling the wireless connections. Each node is assumed to be able to transmit or receive at a fixed rate.

The shared nature of the wireless medium is modeled by introducing constraints on which transmitter-receiver pairs can be active simultaneously. One extreme case of such a network is the *collocated network*, in which each transmission is heard by all other nodes, and only one node can transmit in a single time-slot. Another network of interest is the *random multi-hop network* consisting of nodes uniformly scattered on a unit square, in which all nodes have the minimum common range of transmission so that the network is connected. Random graph analysis [5] shows that in a network of n nodes, the minimum such range for the network to be connected with probability approaching one as the network size grows is $\Theta(\sqrt{\frac{\log n}{n}})$. These networks are illustrated in Figure 2.

Note that such models of the network do not allow for cooperative schemes such as relaying, multiple access, beamforming, etc.

The problem of computing functions over such networks was studied in [4]. A direct generalization of the two-party case would involve determining the minimum time required to compute a certain function of the set of sensor measurements,



(a) (b)

Fig. 2. (a) Collocated network (b) Random network

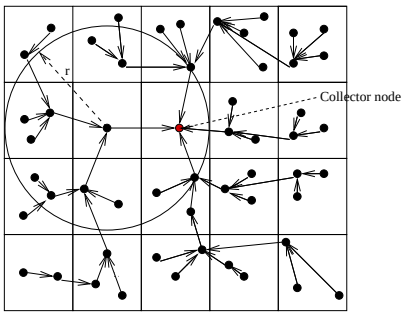


Fig. 3. Forming a tree rooted at the collector

i.e., a *one-shot* computation problem. Instead, we consider the *block computation* problem, in which sensors take periodic measurements $[x_1(t), x_2(t), \ldots, x_n(t)], t = 1, 2, 3, \ldots$ of the environment, and the user requires a certain aggregate function $f(x_1(t), x_2(t), \ldots, x_n(t))$ for each time $t = 1, 2, \ldots$. Sensor network applications which involve constant monitoring of the environment often require such repeated computation.

There are two possible ways to perform such computation. One is to separately compute each function value, which would correspond to the one-shot computation problem. The other way would be to let each sensor accumulate a block of measurements, subsequent to which the entire network performs joint computation of the block of corresponding function values. Such an approach could be potentially more efficient, though at the cost of greater latency because the individual function values become known to the collector only after the entire block is computed. Note that this is a direct analogy of block coding in communication theory.

The measure of efficiency we study in a block-computation setting is what we call *computational throughput*: If a computational scheme requires T time units to compute a block of N function values, the computational throughput is $\frac{N}{T}$. We seek the supremum of the computational throughput over all schemes and all block-lengths N.

Let us consider the class of *symmetric functions*. These are functions which are invariant to permutations of their arguments, i.e., $f(x_1, \ldots, x_n) = f(\pi(x_1, \ldots, x_n))$ for any permutation π. The interest in such functions is twofold. First, most statistical functions of interest including mean, median, maximum/minimum, histogram etc. Also, such functions have some properties which can be exploited, as described below.

The key property of symmetric functions is that they are determined completely by the histogram of the set of node measurements. Further, the histogram itself has two useful characteristics: 1) The histograms of two sets of measurements can be combined to give the histogram of the union, and 2) Where the individual sensor measurements are D-valued, the histogram of n measurements can be represented in $O(D \log n)$ bits.

The first property suggests simple schemes to compute a function in a distributed manner. One could, for instance, form a tree and propagate partial histograms from children to parents up the tree towards the root. As we will see in Section VI, many simple randomized algorithms to compute functions are applicable to functions with this property.

The above properties make it possible to compute any symmetric function with computational throughput $\Omega(\frac{1}{\log n})$ in any network with maximum degree $O(\log n)$. The corresponding computational cycle-time, i.e., the reciprocal of the computational throughput, is exponentially smaller than the computational cycle time of $O(n)$, which is the minimum cycle-time for downloading all data to the collector. The scheme to achieve this rate consists of first constructing a suitable tree, rooted at the collector node, and formed by grouping nodes according to a tessellation of the plane into cells containing approximately $O(\log n)$ nodes each. Such a tessellation for the random network on a square domain is shown in Figure 3. By collecting histograms along this tree, and with appropriate scheduling and pipelining, it is possible to obtain the desired computational throughput of $\Omega(\frac{1}{\log n})$.

Examples of networks for which a computational throughput of $\Omega(\frac{1}{\log n})$ is achievable include the random network on a square (in this case, the throughput is achievable with high probability for large n) as well as constant degree networks such as lattice grids.

Note that $O(\frac{1}{\log n})$ is the maximum possible throughput to communicate the histogram of the set of measurements, due to the fact that representing the histogram itself requires $\Theta(\log n)$ bits, and the collector node can receive only a bounded number of bits per slot. Thus the maximum computational throughput for computing the histogram over networks of maximum degree $O(\log n)$ is $\Theta(\frac{1}{\log n})$.

To obtain similarly sharp results for other symmetric functions, it is necessary to more precisely characterize what computational schemes or protocols are allowable. The simplest scenario to study this problem in a network setting is the collocated network. A suitable generalization of the class of communication complexity protocols described in the last section can be defined for this network as well. Combinatorial arguments can then be employed to lower bound the time required by any such protocol to compute a vector of function values corresponding to a large block of sensor measurements.

In particular, let us consider the class of *type-sensitive functions*, which are functions for which at least a certain

minimum fraction of arguments need to be known for the function value to be determined. Instances of type-sensitive functions include Average, Median, Majority, Histogram, and many others. Counting arguments can be used to show that the maximum rate for computing type-sensitive functions in the collocated network is $O(\frac{1}{n})$. Since all the data can be downloaded to the collector in time $n \log D$, this implies that this particular class of functions is maximally difficult to compute in terms of order.

For a certain subclass of symmetric functions, an exponential speedup is possible. This is the class of *type-threshold* functions, which are defined as functions that depend only on the element-wise maximum of the histogram and a fixed threshold vector. In the case of binary measurements, for instance, a type-threshold function would only depend on whether the number of ones exceeds a fixed threshold number. The canonical example of a type-threshold function is the Max function. In the collocated network, type-threshold functions can be computed with a throughput of $\Omega(\frac{1}{\log n})$. To see how such a protocol can be constructed, consider the case in which the measurements are binary and the Max function is to be computed. The scheme essentially works by each node sequentially communicating the set of entries in its corresponding block with measurement value 1, *except* for those entries for which the function value is already known to be 1, due to the transmissions of previous nodes. Combinatorial arguments show that the total number of bits over all transmissions will be $O(N \log n)$, which translates to a computational throughput of $\Omega(\frac{1}{\log n})$.

It is also possible to show, by generalizing the fooling set arguments mentioned in the last section, that this is order optimal. Thus, the maximum computational throughput for type-threshold functions such as Max is $\Theta(\frac{1}{\log n})$, and is $\Theta(\frac{1}{n})$ for type-sensitive functions such as Average, Median and Histogram.

The arguments for the collocated network can also be extended to the random multi-hop network, to show that with probability approaching one as $n \to \infty$, the maximum computational throughput is $\Theta(\frac{1}{\log n})$ for type-sensitive functions and $\Theta(\frac{1}{\log \log n})$ for type-threshold functions.

One feature to note about the above results is that they are *scaling* results in the *network size*, whereas communication complexity results are often order results in terms of *alphabet size*. The former are thus more applicable when the network size is considerably larger than alphabet size. In a scaling sense, we see that the computational throughput for type-threshold functions is exponentially higher than for type-sensitive functions, in both the collocated network as well as the random multi-hop network. Also, multi-hop networks allow for a far greater degree of in-network compression, and consequently allow a higher computational throughput than the collocated network.

There are however some drawbacks with the model and results described above. First, the model does not incorporate node or link failures. It is not clear if these results easily extend to such unreliable scenarios. Secondly, there may be considerable overhead required for nodes to know what computational operations they must carry out; the "roles" of individual nodes depend on their locations and thus must be dynamically assigned.

A further drawback of the model is that it does not take into account correlations in the source measurements, which could be exploited if the requirement of exact computation with probability one were relaxed. Also, the communication model is a simplification of the more general multi-user noisy channel model which allows arbitrary modes of operation.

IV. Reliable Computation in the Collocated Network with Noise

In wireline networks, the physical layer implements error correction so that communication links appear reliable to the applications. It then becomes possible to view the network as a collection of point-to-point reliable links, for the purpose of one-way communication over the underlying physical layer channels.

But consider now a distributed computation task over a network with noisy links. Is it similarly possible to make the overall computation as reliable as desired? And if so, at what cost?

Classical information theory tells us that reliable communication, i.e., with error probability as low as desired, at a positive throughput, is possible if a large block of data is communicated with appropriate redundancy. By employing sufficient coding overhead and large enough block-length, the physical layer can be made as reliable as desired.

Thus, if the communication in a protocol which assumes noiseless communication is composed of sufficiently long blocks of information bits transmitted from node to node, error correction can be employed to provide reliability of each link, at the cost of some overhead. Such a protocol can therefore be converted to a protocol which functions reliably over a noisy network with a constant factor overhead. For the block-computation case, it is not hard to show that this can be done, at least for the cases described in the previous section. The scaling results would thus remain the same for a noisy communication model, although the latency would be high due to the long block lengths necessary for reliability.

However, classical information theory does not answer the question of how a general *interactive* protocol designed for a noise-free network can be made reliable in the presence of noise. Consider a large network of several nodes in which a distributed computation task consisting of several point-to-point transmissions and computations is to be performed. How much error correction overhead is needed to make the overall computation reliable? The naive approach would be to make each transmission of the interactive protocol sufficiently reliable, possibly through repetition coding, so that the union of all the error events on all the transmissions in the protocol has sufficiently low probability. In networking terms, this would translate to providing link level reliability for every transmitted packet. While this naive strategy cannot be dismissed out of

hand, the larger the number of transmissions in the interactive protocol, and the smaller the number of bits in each individual transmission, the greater the overhead of such a scheme.

One of the simplest such problems was posed by El Gamal, and studied by Gallager [6]. It involves a collocated network consisting of n nodes, each of which stores a one-bit measurement. The communication model is broadcast with binary symmetric errors independent from receiver to receiver. That is, only one node can transmit in a time slot, each transmission is a single bit, and each node independently receives the transmitted bit or its complement with probabilities ϵ and $1-\epsilon$ respectively. Consider the problem where the collector node desires to determine the *parity* of the total number of 1's in the network.

The problem is to minimize the total number of transmitted bits which will guarantee that the parity will be known to within a desired probability of error. Note that this relaxed requirement of inexact computation is necessitated by the noise in the channel; exact computation with probability one is impossible.

This is a simple instance of a problem in which the information quantity at each node is small, but the total information distributed among many nodes is large. In the noiseless case, it is easily seen that n transmissions are required to compute the parity function exactly. This is thus a lower bound for the noisy case.

The key feature of this problem is that any broadcast is heard in independent noise by n nodes, which could collectively thus make a good estimate of the bit. However, to make this pooled estimate known collectively would consume several transmissions. One could consider the naive approach of repetitive coding, which consists of each node transmitting its own bit k times, resulting in a total of kn transmissions. The receiver could make a maximum likelihood decision on the bit of each node. Simple analysis shows that if the desired probability of error is sufficiently low, the number of transmissions per node, k, must grow as $\log n$, which requires a total of $\Theta(n \log n)$ transmissions. But this approach does not utilize the broadcast nature of the receptions at all.

Gallager's contribution was to devise a scheme which requires only $O(n \log \log \frac{n}{\delta})$ transmissions to guarantee a probability of error in the computation of parity of less than δ. The scheme consists of dividing the nodes into subsets of size $\Theta(\log n)$. There are two phases in the algorithm. In the first phase, nodes transmit their values repeatedly k times, followed by each node estimating the parity of the sum of bits in its own subset. In the second phase, each node transmits its estimate exactly once, after which the collector makes a maximum likelihood decision. Analysis shows that choosing $k = \Theta(\log \log \frac{n}{\delta})$ guarantees a probability of error less than δ.

It is further shown that by using a suitable modification of this scheme, with a number of transmissions per node that is of the same order $k = \Theta(\log \log \frac{n}{\delta})$, all the nodal bit values can be obtained by the collector with high probability. Whether $\Omega(n \log \log n)$ transmissions is also necessary is still an open question.

The same collocated network and noise model was also studied by Kushilevitz and Mansour [7]. They considered the class of *threshold* functions, i.e., functions which are determined by whether the number of 1's in the network exceeds some threshold (in contrast to the class of *type-threshold* functions mentioned in the previous section which required that the threshold be fixed and independent of n). These include the And, Or, and Majority. For this class of functions, they showed how to construct a protocol which computes the function in time $O(n)$. This is order optimal, because the number of transmissions required to compute any threshold function in the noiseless case is n (for a proof of this elementary result see [4]).

An interesting property to note about the achievable schemes for the two problems described above is that they are *oblivious*, i.e., the order of transmission does not depend on previous transmissions. This is not true of many of the non-trivial protocols devised for the problems in the noise-free model, for example median computation in the two-party communication complexity case (see [2] for a description of this protocol), and the Max computation in the collocated network.

The direct applicability of these results relies on a scenario where a large number of sensors are physically proximate enough so that each transmission is a broadcast, which may not be very realistic. However, this formulation is definitely worth studying because that it captures the shared nature of the wireless medium as well as the cooperative nature of the distributed computation problem, which are the two essential features of the overall problem of computing over wireless networks.

V. Interactive Communication in Noisy Networks

The previous sections have addressed distributed computation problems which are posed as functions of distributed data that are required to be communicated to a certain collector. These are special cases of a more general notion of an *interactive protocol*, which was introduced by Schulman in a two-party setting [8], and subsequently generalized to arbitrary networks by Schulman and Rajagopalan [9].

The notion of an interactive protocol is quite general and powerful. Given a network of processors connected by communication links, it can be defined as follows: Each processor contains some input value, and the protocol specifies a set of transmissions which terminate after time T, where each transmission made by a processor on an outgoing link is a function of its input and previously received transmissions. The objective of such a protocol may be communication of a function, or simply data communication from sources to some other nodes. Regardless, the only assumption that need be made is that the protocol has available noiseless information links for its operation. Figure 4 shows the operation of such a protocol at a single node.

Schulman studied the problem of "simulating" such protocols on networks with noise. By "simulation" is meant the

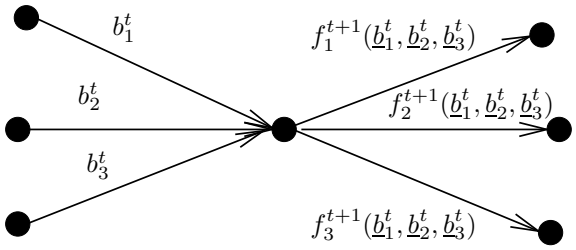


Fig. 4. Operation of an interactive protocol at a single node

following: Given the input values available initially to each node, specify a protocol on the noisy network, i.e., for each node and each time slot, a mapping, from past received bits (possibly in error) and input, to bits to be transmitted on the outgoing links, such that after this protocol terminates, each node can reconstruct an exact copy of the sequence of transmissions and receptions in the original noiseless protocol.

As in the one-shot communication case, classical information-theoretic coding theorems are inapplicable. This is because a node does not have access to the block of information it wishes to transmit; this information becomes available, only bit by bit, as it receives transmissions from other nodes. Note further that these received transmissions may be corrupted by noise. Thus, errors could be due not only to incorrectly received bits, but also due to wrong transmissions made due to past erroneous receptions.

In [8], this problem is considered for a two node network in which the nodes are connected by a pair of binary symmetric channels in each direction, each with capacity C. It is then shown that any noiseless protocol that requires T time-slots can be simulated on the noisy network by a protocol which requires time $O(\frac{T}{C})$, with error probability $e^{-\Omega(T)}$.

In [9], this result is extended to a network setting, where a network of n nodes are linked together arbitrarily by binary symmetric channels of capacity C bits/slot. If d is the maximum degree of any node in the network, it is shown that any noiseless protocol which requires time T can be simulated by a noisy protocol which requires time $O(\frac{T}{C}\log(d+1)+\frac{1}{C}\log n)$.

The technique that is introduced and used in the simulation scheme is the concept of a *deterministic tree code*. This consists of a labeled $k-$ary tree in which the edges are labeled with code symbols belonging to a certain alphabet. A string of $k-$valued information symbols can be mapped to a code-word as follows: Trace the path specified by the string down the tree, where at each level the next information symbol specifies which of the k children of the current node to go to next. The code-word is then the sequence of labels on the edges of the path traced. Note that this mapping need not be done with complete knowledge of the input stream a-priori; input symbols can become known one at a time.

The required property of such a tree code is that given any two nodes at the same depth d', with lowest common parent at depth h, the Hamming distance, or number of differing entries, in the corresponding codewords is at least $\frac{1}{k}(d'-h)$. It can be shown [8] that a edge-labeling alphabet of fixed finite size suffices for a tree of arbitrary depth. An information symbol is mapped to a tree code symbol by locating the node corresponding to the string of previous transmissions, and identifying one of the k edges corresponding to the information symbol. Decoding of a received symbol is done by concatenating it with all the past received symbols and determining the corresponding maximum likelihood path in the tree.

The structure of a tree code guarantees that the greater the number of receptions after a certain received symbol, the greater the degree of confidence in decoding it correctly.

The overall scheme consists of two layers. At the higher layer, each node decides whether its past receptions and transmissions are "consistent" with the original noiseless protocol, and if so to transmit a symbol specified by that protocol. If, on the other hand, it discovers a mismatch between the currently decoded sequence of past transmissions and some previously decoded sequence, it decides to transmit a "backup" symbol, which is a notification of error. These higher level symbols are translated to actual bits via the tree code. As more and more symbols are received, nodes will be able to correct past errors and subsequently propagate these corrections.

One difficulty in the above results is that while tree codes can be shown to exist, explicitly constructing them remains an open problem. Also, the question of how to construct a good noiseless protocol to carry out a particular computation is not addressed.

From the perspective of sensor networks, a better formulation for computing under uncertainty may involve error events such as packet drops and node failures, rather than bit level error events. The relevance of the latter goes back to the discussion in the beginning of Section IV, and may be restricted to protocols which are "large" enough in terms of data so that the overhead of guaranteeing overall computational reliability through per-packet reliability becomes too large.

VI. Information Theoretic Formulation

Distributed computation problems can be posed in more general information theoretic settings. This permits the addition of a feature that is absent in all the previous formulations - the possible exploitation of correlation in the sensor measurements, which could create a great deal of redundancy. We now review some of the basic information theoretic results, which lead to a more general formulation that incorporates computation over wireless networks. We should note that currently there are virtually no results in information theory encompassing all these elements. So this remains a program for the future.

We begin by considering the simple case of two sensors, which take measurements in each time slot that are jointly correlated, but temporally independent and identically distributed. The distributions of each pair of measurements of the two sensors are given by two random variables X and Y, which have a certain joint distribution (see Figure 5(a)). The two sensors are connected through noiseless independent links to a receiver, to which the measurements need to be communicated.

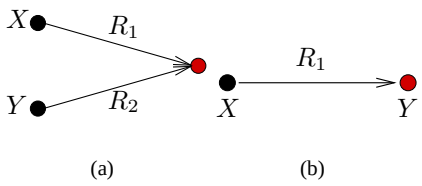


Fig. 5. (a) Slepian-Wolf problem. (b) Wyner-Ziv side information problem.

The question of interest is to determine the *rate region*, or the set of all possible pairs of rates at which the sources can be individually compressed and sent to the receiver, such that the the receiver can reconstruct the original sources with vanishing probability of error. This problem of *decentralized compression of correlated sources* was considered by Slepian and Wolf [10].

The challenge in this problem is to exploit the correlation in the measurements, even though the two sensors are separated. If in fact the two sensors did have complete access to each others' measurements, then the set of rates which would allow perfect source reconstruction would in fact be all pairs R_1, R_2 satisfying $R_1 + R_2 \geq H(X,Y)$, where $H(X,Y)$ is the joint entropy of the random variables X and Y[2]. Slepian and Wolf proved the remarkable result that this same rate region, truncated by the additional constraints $R_1 \geq H(X/Y)$ and $R_2 \geq H(Y/X)$, is achievable even when the sensors have no access to each other's measurements. In particular, this means that if $R_1 \geq H(X/Y)$, then the required rate for the second source is the same in the no-access case as it is in the complete access case (and the same is true for the first source when $R_2 \geq H(Y/X)$.

In a wireless sensor network setting, it may be more appropriate to replace the independent links with a *multiple access* channel, and pose the feasibility question in terms of what transmit power pairs would allow the sources to be communicated. This problem is open for the general case of arbitrary joint distributions.

Another variation of the problem would arise if what is desired is not perfect communication of the sources, but rather reconstruction of the sources to some *fidelity*, i.e., the receiver wishes to recover estimates $\hat{X}, \hat{Y}$ such that $E[D(X,\hat{X})] \leq d$, $E[D(Y,\hat{Y})] \leq d'$, where $D(\cdot,\cdot)$ is a given *distortion measure*. This problem is open as well.

The special case of this problem in which one of the sources is known to the receiver as *side information* (see Figure 5(b)), and only the other is to be determined, was solved by Wyner and Ziv [11]. Orlitsky and Roche [12] have extended this to the case in which the receiver desires to know a certain function $F(X,Y)$ of the single source X and the side information Y, and determined the required capacity of the channel between the source and receiver as being a function of the *conditional graph entropy*, which is a measure defined on the two random variables and a certain graph defined by the function $F(\cdot)$.

Another problem of interest in a wireless setting would be a generalization of the formulation in [4] to a more general channel model. Specifically, consider the problem of communicating a function of independent sources to a receiver, where the sources have access to a multiple access channel. One of the simplest examples of such a problem is the following: Consider two sources S_1 and S_2, which have access to channel inputs X and Y of a multiple access channel, with output $Z = X + Y + N$ being available to a receiver, where N is Gaussian noise. The receiver desires to know the sum $X+Y$. The question of interest is to find the optimal power-distortion curve, i.e., for a given pair of transmit powers P_1 and P_2, what is the minimum distortion D at which the sum $X+Y$ can be communicated to the receiver.

One could go further, and ask whether a set of correlated sources could be communicated over a wireless multi-hop network at a desired level of fidelity with respect to a certain joint distortion criterion, which would model the function to be computed. Such a joint distortion criterion could, for example, be defined as $D((X,Y),(\hat{X},\hat{Y})) := |f(X,Y) - f(\hat{X},\hat{Y})|^2$. This would model a very general version of the problem of distributed computation of a function over a multi-hop wireless network. The solution to such a problem is unfortunately very far from the current frontiers of what is known in information theory.

VII. Gossip Based Aggregation in Networks

The formulations and results considered so far can be classified roughly as being concerned with determining the optimal or at least order-optimal scheme to compute a desired aggregate function of nodal values. Some of these problems allow for channel errors. However, most are static solutions in that they do not account for nodal or link failures or changes in network topology. In addition, such schemes require nodes to know how to process and route data, which is larger in terms of overhead than, say, routing protocols. In networks in which such nodal/link failures are frequent, such as *peer-to-peer* as well as sensor networks, *randomized gossip algorithms* have been studied as possible ways of providing robustness and fault tolerance, as well as scalability.

Gossip algorithms essentially consist of information propagation through nodes randomly selecting neighbors to transmit to in each round. There has been considerable work on constructing gossip algorithms to compute aggregate functions in networks [13, 14]. In comparison to the work described in the previous sections, such approaches are more limited in that the functions considered in most cases are limited to averages, sums and extremal values. The advantage however is fault tolerance as well as simplicity in implementation; the computational operations that nodes have to perform are restricted to very simple ones.

Consider a network in which nodes initially contain some *state value*. In each round, each node can transmit a single packet containing an arbitrary value to any other node. There

[2]More precisely, blocks of measurements could be communicated at the corresponding rates with probability of error going to zero with increasing blocklength.

is no explicit cardinality bound on the number of values that such messages can take, but the implicit assumption is that the message size is large enough to permit a high enough degree of precision.

Typically, gossip algorithms to compute aggregates work by maintaining one or more state values, and updating the state value in each round after transmitting/receiving a certain number of messages from randomly chosen neighbor(s). For instance, [13] proposes a simple *push* algorithm to compute the average. The constraint in a push algorithm is that a node can receive multiple messages in a single round, but can transmit only one message to one other node. The algorithm works as follows: Each node maintains two state values, *sum* and *weight*. In each round, a node sends $\frac{sum}{2}$ and $\frac{weight}{2}$ to another randomly chosen neighbor, and to itself. It then performs element-wise addition of all the received pairs to obtain an updated pair of state values. The estimate of the average is $\frac{sum}{weight}$.

The problem that gossip algorithms which compute aggregates must deal with is the possibility of double counting, i.e., contacting the same node more than once. The above algorithm maintains the consistency property that the sum of all the node weights is n, and the average of all the sums is the true average. As a consequence of this, double counting is not an issue. It is further shown in [13] that the above push algorithm requires $O(\log n + \log \frac{1}{\epsilon} + \log \frac{1}{\delta})$ rounds to guarantee an ϵ approximation with probability or error less than δ. This is order optimal, since it can be shown [15] that any gossip algorithm in any n node graph requires $\Omega(n)$ rounds to compute the average with probability of error less than ϵ, where $0 < \epsilon < 0.5$.

This algorithm is further generalized to compute functions based on *linear synopses*. Linear synopses are functions f on multi-sets which satisfy $f(S_1 \bigcup S_2) = f(S_1) + f(S_2)$, i.e., the function values on distinct subsets of nodes can be additively combined to give the function value of the union.

In addition, the performance of these algorithms is analyzed with respect to random node and message failures, and is shown to have a constant factor performance loss.

It is assumed in [13] that the network topology is that of a complete graphs; in each round, any node can transmit to any other node. The problem of computing the average over general graph topologies is considered in [14]. A class of randomized algorithms characterized by matrices with a doubly stochastic structure is proposed. These algorithms can be defined for arbitrary connected graphs. The performance of these algorithms is shown to depend on the second largest eigenvalue of the corresponding matrices. An application of the main result of [14] to the random graph on a plane with common transmission range r shows that $O(\frac{\log n}{r^2})$ rounds suffice to obtain a probability of error of less than $\frac{1}{n}$.

The advantage of such algorithms, as mentioned earlier, is their simplicity and fault tolerance. The main limitation is that only a restricted class of functions is considered. Another drawback is that the constraints on per-round communication that are typically assumed in gossip algorithms do not match with the constraints in wireless networks. For example, in a complete graph, the gossip formulation allows every node to transmit/receive in a single round, whereas in a wireless setting this would not be possible. In order to properly evaluate gossip algorithm performance vs. the limits of what is possible in a wireless network, it is necessary to account for these differences in constraints, and furthermore account for how message size, which for the gossip algorithms is assumed to be fixed, grows with the number of nodes.

VIII. Conclusions

A theory of distributed computation in sensor networks is still in its infancy, if it exists at all. Many of the formulations described in this paper apply to general networks of processors connected by communication links, and as a consequence do not capture all the essential features specific to wireless sensor networks. None of the approaches address the essential problem of how to compute a general function with input distributed over a network, in a way that is robust to unreliability, including link failures, packet drops and node failures, and is scalable. Upper bounds to performance are particularly lacking in such problems. While it is often not hard to construct schemes for specific functions, what is also needed is a guarantee on how far from optimal such schemes are.

Aggregate functions with relatively simple structure such as symmetric functions are somewhat easier to handle, but many functions of interest do not have such a symmetric structure, an important instance being location dependent functions. Thus, an approach for a more general class of functions is also needed.

A comprehensive theory also needs to address energy consumption and network lifetime, in addition to latency and throughput, as metrics of performance. There is much yet to be done.

References

[1] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TinyDB: An acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, 2004.

[2] E. Kushilevitz and N. Nisan, *Communication Complexity*. Cambridge University Press, 1997.

[3] P. Tiwari, "Lower bounds on communication complexity in distributed computer networks," *J. ACM*, vol. 34, no. 4, pp. 921–938, 1987.

[4] A. Giridhar and P. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas of Communication*, vol. 23, no. 4, pp. 755–764, April 2005.

[5] P. Gupta and P. Kumar", ""critical power for asymptotic connectivity in wireless networks"," in *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming*, W. McEneaney, G. Yin, and Q. Zhang, Eds., 1998.

[6] R. Gallager, "Finding parity in a simple broadcast network," *IEEE Trans. Inform. Th.*, vol. 34, no. 2, pp. 176–180, March 1988.

[7] E. Kushilevitz and Y. Mansour, "Computation in noisy radio networks," in *SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1998, pp. 236–243.

[8] L. J. Schulman, "Coding for interactive communication," *IEEE Trans. Inform. Th.*, vol. 42, no. 6, pp. 1745–1756, Nov. 1996.

[9] S. Rajagopalan and L. Schulman, "A coding theorem for distributed computation," in *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM Press, 1994, pp. 790–799.

[10] D. Slepian and J. Wolf, “Noiseless coding of correlated information sources,” *IEEE Trans. Inform. Th.*, vol. 19, no. 4, pp. 471–480, July 1973.

[11] A. Wyner and J. Ziv, “The rate distortion function for source coding with side information at the receiver,” *IEEE Trans. Inform. Th.*, vol. IT-22, pp. 1–11, 1976.

[12] A. Orlitsky and J. R. Roche, “Coding for computing,” in *IEEE Symposium on Foundations of Computer Science*, 1995, pp. 502–511.

[13] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based computation of aggregate information,” in *FOCS ’03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2003, p. 482.

[14] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Gossip algorithms: Design, analysis and applications,” in *Proceedings of IEEE Infocom 2005*, 2005.

[15] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, “Randomized rumor spreading,” in *FOCS ’00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2000, p. 565.