# Collecting Correlated Information from a Sensor Network

Micah Adler*
Department of Computer Science
University of Massachusetts, Amherst
Email: micah@cs.umass.edu

August 3, 2004

**Abstract**

A fundamental problem in the study of sensor networks is collecting data to a central server from a set of $k$ distributed sensor nodes. A considerable amount of recent research in this area attempts to reduce the number of bits sent by the nodes by taking advantage of correlations between the data items collected from different nodes. In this paper, we study this problem in the following model: let $D$ be a probability distribution over $k$ binary strings of length $n$. $D$ is given to the server but not the nodes. A sample $\bar{X}$ is drawn from $D$ and the $k$ strings of $\bar{X}$ are revealed to the nodes, one string per node. The goal is to inform the server of all $k$ strings of $\bar{X}$. Our primary objective is to minimize the total number of bits sent by the nodes, but we also seek to minimize both the bits sent by the server and the number of rounds required. This problem is a natural parallelization of the model introduced in [2], and is also well motivated by recent work on distributed source coding for sensor networks. Our main result is a protocol that allows the server to correctly determine $\bar{X}$. In this protocol, the nodes send $O(H(D)+k)$ bits in expectation, where $H(D)$ is the binary entropy of $D$; this is asymptotically optimal. The server sends $O(kn + H(D)\log n)$ bits in expectation, and the number of rounds is $O(1+\log\lceil H(D)\rceil)$ in expectation. We also demonstrate that if the server is allowed to produce an incorrect result with probability up to $\Delta$, then the expected number of bits sent by the server can be reduced to $O(k\log(\frac{nk}{\Delta}) + H(D)\log n)$, without increasing the other measures of performance.

## 1 Introduction

We consider the problem of collecting information to a central server from a set of $k$ distributed nodes. In particular, we assume that each of the $k$ nodes has an $n$-bit string to send to the server, and these strings are correlated in some manner. We denote such a set of strings as $X = (x_1, x_2, \ldots, x_k)$, where $x_j$ is the $n$-bit string known to node $j$. We assume $X$ is drawn from a probability distribution $D$ over such sets, where $D(X)$ denotes the probability that $X$ is drawn from $D$. Note that $D$ captures both the *a priori* information known about what strings each node is likely to hold, as well as the correlations between the strings given to different nodes.

The specific problem we study is defined as follows: the distribution $D$ is revealed to the server but not the nodes. A sample $\bar{X}$ is drawn from $D$ and the $k$ strings of $\bar{X}$ are revealed to the nodes, one string per node. The server starts with no information about $\bar{X}$ other than the distribution $D$; the objective is to inform the server of all $k$ strings of $\bar{X}$. Communication proceeds in rounds as in

*Supported by NSF Research Infrastructure Award EIA-0080119, NSF Faculty Early Career Development Award CCR-0133664, and NSF ITR Grant (ITR-0325726).

Yao's communication complexity model [27]. In the first half of each round, the server sends bits to the nodes, where each such bit is received by a single node specified by the server. In the second half of each round, the nodes send bits back to the server. The nodes are not able to communicate directly with each other. Our primary objective is to minimize the total number of bits sent by the nodes. We are also concerned with the total number of bits sent by the server, as well as the total number of rounds required.

The motivation for this problem is two-fold. First, it is a natural parallelization of the asymmetric communication problem introduced in [2], and studied subsequently in [13], [12], [24], and [25]. That work deals with the version of the problem where there is only a single node (i.e., $k = 1$), and has led to a number of different protocols for informing the server of the node's $n$-bit string, where the expected number of bits sent by the node is $O(H(D)+1)$, and the expected number of bits sent by the server is $O(n)$. Here, $H(D)$ is the binary entropy of $D$. These protocols vary on a number of measures, such as the number of rounds required and the constants hidden in the asymptotic notation. Techniques from these protocols have been shown to be useful in circumventing web censorship and surveillance [8], as well as in the design of websites [4].

Second, our problem addresses the main difficulty of a fundamental issue in sensor networks: collecting correlated information distributed across a set of sensor nodes to a central server. Information collected by different nodes in a sensor network can be correlated in a number of different scenarios, such as measuring weather conditions in a geographic region or sensing the visual image of an object from similar but distinct viewpoints. Furthermore, sensor nodes are often quite limited in terms of both available power and computational capabilities. This motivates our primary objective: since wireless transmission is a power intensive operation, we wish to minimize the number of bits sent by the nodes. Furthermore, the limited computational capabilities of the sensor nodes motivate the assumption that the nodes do not have *a priori* access to the distribution $D$.

This sensor networks application has generated considerable research in the last few years on what is known as distributed source coding (DSC). DSC dates back to the seminal work of Slepian and Wolf in the early 70s [21]. Stated in terms of our model, the objective of a DSC is to send a sequence of $r$ independent samples drawn from the probability distribution $D$ to the server, without the nodes receiving any feedback from the server (and, as in our model, without communicating with each other). Slepian and Wolf demonstrate in [21] that there exists an encoding scheme such that as $r$ grows, the number of bits that must be sent to the server approaches $rH(D)$, which is optimal. Since this result is asymptotic (in the sense that $r$ must grow without bound), as well as not constructive, there has been intense study recently of more practical DSC techniques. We provide a brief description of this work in Section 1.1, and refer the reader to the survey in [26] for more information (the bibliography of the Web version in particular is quite extensive and recent).

We here point out that all existing work on DSC makes one of two assumptions: either the correlation between the strings has a restrictive structure[1] or the required value of $r$ is large[2]. Note that a value of $r > 1$ can also be viewed as the case where $r = 1$ with a significant restriction on the distribution $D$: the strings can all be partitioned into equal length substrings, each of which is *i.i.d.* The survey [26] describes generalizing the correlation model as the main issue for practical deployment of DSC to sensor networks, and states that this is "still an open and very challenging DSC problem." In this paper, we introduce a protocol that works for **any** distribution $D$ with $r = 1$, and thus for any correlation model. Note, however, that the communication model we consider in

[1]For example, in the case where $k = 2$, both strings may be uniformly distributed, and the correlation model is that there is a bound on the Hamming distance between them.

[2]Typical values of $r$ described in [26] are $10^4$ and $10^5$, and [14] describes experiments where $r = 5 \times 10^5$.

this paper is more powerful than the classical DSC model of [21], as we allow interaction where the server sends information back to the nodes. Such feedback has already been utilized in more recent work on DSC, for example in [5].

This kind of interactive communication is also used for the case where $k = 1$ in [2]. In fact, a protocol from [2] called **Round-Efficient** could be used to solve the problem we consider in this paper using two different methods, both of which are suboptimal. In the first method, **Round-Efficient** is used on each of the nodes independently and in parallel. This would require $O(nk)$ bits to be sent by the server and $O(1)$ rounds (both in expectation). As we describe below, $\Omega(nk)$ total bits must be exchanged, and thus both of these measures are optimal. However, if $D$ has a large amount of correlation between the $k$ strings, the number of bits sent by the nodes can be as large $\Omega(kH(D))$, a factor of $k$ larger than optimal. In the second method, **Round-Efficient** is used in a *serial* fashion: the server first collects the string from one node, conditions on the value it receives, and then moves on to the next node, until it has all $k$ strings. The expected number of bits sent by the nodes in this protocol is $O(H(D) + k)$ (by the definition of conditional entropy) and the expected number of server bits is $O(nk)$, both of which are asymptotically optimal. However, the expected number of rounds required is $O(k)$, which for large $k$ would be quite time consuming.

Our main result is to provide a new technique for this problem that is significantly more parallel than such a serial application of **Round-Efficient**, while requiring roughly the same number of bits to be sent by both the server and the nodes. In particular, we introduce a protocol that solves the information collection problem, where the expected number of bits sent by the nodes is $O(H(D)+k)$, the expected number of bits sent by the server is $O(nk + H(D) \log n)$, and the expected number of rounds is $O(1 + \log\lceil H(D)\rceil)$. This protocol applies to **any** probability distribution $D$ (and hence any correlation structure) over the $k$ strings of length $n$.

To construct this protocol for general probability distributions, we first consider a restricted class of distributions $D$. In particular, we consider *support uniform* distributions: distributions that are uniform over their support set. For a support uniform distribution $D$, let $\chi(D)$ be the support set, i.e., the set of possible inputs $X$ such that $D(X) > 0$. This special case is actually a natural parallel communication complexity problem. We can think of the distribution $D$ as a $k$-dimensional binary matrix, with side length $2^n$, where the 1 entries of the matrix represent the elements of $\chi(D)$. The server is given this matrix. An entry of the matrix containing a 1 is chosen uniformly at random, and node $i$ is given the $i$th coordinate of this entry. The objective is for the server to determine which entry of the matrix has been chosen. We provide a protocol for the case of support uniform distributions such that the expected number of bits sent by the nodes is $O(m + k)$, where $m = \lceil \log |\chi(D)| \rceil$ (and thus the performance could also be stated as $O(H(D) + k)$). The expected number of server bits is $O(nk + m \log n)$, and the expected number of rounds is $O(1 + \log m)$. We then demonstrate that this protocol can be adapted to the case of an arbitrary probability distribution.

Shannon's Theorem implies that in our protocol for arbitrary distributions, the expected number of bits sent by the nodes is asymptotically optimal. Furthermore, Theorem 5 of [2] implies that if the server must determine the set of strings held by the nodes with no probability of error, then at least $nk$ bits must be exchanged between the server and the nodes. Our protocol requires $O(nk + H(D) \log n)$ expected server bits. The $O(nk)$ term is thus asymptotically optimal for an error-free protocol that is efficient in terms of bits sent by the nodes. The $H(D) \log n$ term, on the other hand, could potentially be improved, but since the problem we consider is of most interest when $H(D) \ll kn$, this term will typically be smaller than the $O(nk)$ term.

We also demonstrate that if we allow the server some probability of returning the wrong result, then we can reduce the expected number of server bits. In particular, we show that if the probability of making an error can be as large as $\Delta$, for any $0 < \Delta < 1$, then the expected number of bits sent by the server can be reduced to $O(k \log(\frac{kn}{\Delta}) + H(D) \log n)$, without effecting (asymptotically) the number of bits sent by the nodes or the number of rounds. Reducing the expected number of server bits is an important consideration for the application of these techniques to sensor networks, since in some scenarios (see for example [9]), receiving bits at a mobile node consumes only a small constant factor less power than sending bits from a mobile node.

### 1.1 Previous Work

Although a full description of the current state of DSC is well beyond the scope of this paper, we here provide a flavor of this work. Perhaps the best studied case of DSC is for the binary symmetric channel (BSC), and its variants. In this model, each sample drawn from $D$ is a pair of correlated binary Bernoulli(0.5) random variables $\bar{x_1}$ and $\bar{x_2}$, and $\Pr[\bar{x_1} \neq \bar{x_2}] = p$, for $p < \frac{1}{2}$. Note that in this case $n = 1$ and $k = 2$. A number of different schemes for sending a sequence of $r$ independent such samples using close to $rH(D)$ bits have been proposed, including [10, 3, 1, 15, 16]. The case of two binary sources (again, $k = 2$ and $n = 1$) where the correlation between the two bits can be arbitrary was considered in [20]. The case where the number of nodes is larger than two has been considered in [20, 23, 17, 14]. The case of uniformly distributed $M$-ary bits has been studied in [14]. Memory between the different samples (which is closer to our model of arbitrary correlation with $r = 1$) is considered in [11]. That work considers the case of $k = 2$, where the sequence of bits for one node is *i.i.d.* and uniform, and the second sequence of bits is correlated via a hidden Markov model. They do not provide analytic bounds on the relationship between the amount of memory present and the value of $r$ required by their technique, but describe simulation results for a Markov model with two states and $r = 16384$.

Another consideration for this line of research is how many bits each of the nodes sends (instead of just the total number of bits). The original proof of Slepian and Wolf [21] demonstrated that efficient DSC is possible for **any** partition of the bits that satisfies the natural conditional entropy requirements. This generality has been matched by some of the more practical schemes. For example, [20] demonstrates the analogous result for two binary sources (i.e., $n = 1$) and arbitrary correlation between the pair of bits. This is an aspect of the problem that is still relevant for the problem we consider in this paper. Refining our techniques so that the $O(H(D))$ bits sent by the nodes can be divided up arbitrarily (subject to the conditional entropy constraints) is an interesting open problem.

A number of papers have also considered the case of correlated continuous distribution. This is motivated by applications such as a network of sensors measuring the temperature. In this research, the correlation is often modeled by assuming that the value given to a node is a "noisy" version of the value given to other nodes. For example, [19] considers the case of two nodes, where $\bar{x_2} = \bar{x_1} + N$, where $\bar{x_1}$ and $N$ are both zero mean Gaussian random variables. In this area, it is also usually assumed that the nodes are distributed over a geographic area, and the amount of correlation between a pair of nodes depends on the distance between them. In [18], it is assumed that the nodes are distributed over a continuous two-dimensional field; they study the effect of increasing the sensor density to infinity. A linear predictive model is considered in [5].

Another area of work in the field of collecting correlated information from sensor networks (again, a full description of this work is beyond the scope of this paper) has focused on incorporating the

routing structure of the sensor network into the compression technique. See for example [7, 22, 18]. This research models the fact that most sensor networks are expected to require multi-hop routing through the sensor network to send the data to the server. Incorporating such considerations into the techniques described in this paper is another interesting open problem.

## 2 Intuition for and description of a support uniform algorithm

In the remainder of this paper, we use $X$ to refer to a generic possible input, and $\bar{X}$ to refer to the actual input drawn from $D$ and given to the nodes. Also, lowercase variables, such as $x_j$, $x'_j$, or $\bar{x_j}$, refer to a string given to node $j$ on the corresponding (uppercase) inputs, such as $X$, $X'$, or $\bar{X}$, respectively.

We start by giving some intuition for our algorithm for the support uniform case. A key tool we use in this algorithm is what we call a *fingerprint bit* function. This is a class of functions that map an $n$-bit string to a single bit such that if $f$ is chosen uniformly at random from the class, then for any $y_1 \neq y_2$, $\Pr[f(y_1) = f(y_2)] \leq p$, for some $p > \frac{1}{2}$. Standard techniques allow such a class of functions to be constructed for any constant $p > \frac{1}{2}$, where the size of the class of functions is $n^{O(1)}$. Thus, a randomly chosen function from the class can be described using $O(\log n)$ bits. Furthermore, such a description allows even a sensor node with limited computational abilities to easily compute the resulting fingerprint bit. Since the value of the constant $p$ only effects constants we hide with asymptotic notation, we here assume $p = \frac{2}{3}$.

We start with a naive attempt at a solution for the support uniform case that will help highlight some of the difficulties we shall encounter. Recall that $\chi(D)$ is the support set for the distribution $D$, and $m = \lceil \log |\chi(D)| \rceil$. In this solution, in each round, the server chooses $m$ fingerprint functions independently and uniformly at random. The server sends to each node a description of $m/k$ of these functions. Each node $j$ applies these functions to the string $\bar{x_j}$ that it has, and returns the resulting fingerprint bits to the server. This process is repeated until there is only a single element of the support set $\chi(D)$ that is consistent with all of the fingerprint bits received by the server.

Note that in the case where $k = 1$, after the only node present has sent $cm$ fingerprint bits, the expected number of incorrect inputs that agree on all fingerprint bits is at most $(|\chi(D)| - 1)(\frac{2}{3})^{cm}$. This is because any incorrect input $X$ disagrees with any given fingerprint with probability at least $\frac{1}{3}$, independently of all previous fingerprint bits, and thus after $cm$ fingerprint bits, any $X \neq \bar{X}$ has a probability of $(\frac{2}{3})^{cm}$ of agreeing on all fingerprint bits. The result then follows via a union bound over all $|\chi(D)| - 1$ incorrect inputs. Thus, $O(m)$ bits are sufficient using this technique when $k = 1$.

On the other hand, in the case where $k > 1$, a fingerprint bit sent by a node $j$ can only help eliminate an incorrect input $X \neq \bar{X}$ if $x_j \neq \bar{x_j}$. Thus, for example, if for all $j > 1$ and all $X \in \chi(D)$, $x_j$ is the same, then only node 1 has any information about which input has been drawn from the distribution $D$, and so only the bits sent by that node can eliminate incorrect inputs. As a result, this first naive technique can require the nodes to send as much as a factor of $k$ times more than the optimal number of bits.

Ideally, we would like each fingerprint bit sent by the nodes to eliminate at least a constant fraction of the remaining elements of $\chi(D)$ (in expectation). We shall informally refer to a bit with this property as *useful*; a more precise definition is provided below. Our objective will be to obtain many useful bits from the nodes without receiving too many bits that are not useful. However, whether or not a fingerprint bit sent by a specific node $j$ is useful depends on several factors, including the

structure of the set $\chi(D)$, as well as the value of $\bar{X}$. Thus, the server will not know if a fingerprint bit sent by node $s$ will be useful or not, since it does not yet know $\bar{X}$. For example, if all but one element of $\chi(D)$ has $x_j = 0$, and one element has $x_j = 1$, then usually a fingerprint bit sent by node $s$ will not be useful, but occasionally it will be (very) useful.

Note that whether or not a fingerprint bit is useful depends on the elements of $\chi(D)$ that have not been eliminated by earlier fingerprint bits. Since many fingerprint bits must be sent in parallel to be efficient in terms of rounds (both from the same node and from different nodes) the decision about how many fingerprint bits a node sends must be made before it is known whether or not those bits will be useful. This is because bits sent in parallel by other nodes may effect whether they are or are not useful. This is perhaps the most difficult aspect of determining whether a fingerprint bit will be useful.

Despite these difficulties, the server can determine a property on a fingerprint bit that ensures that the expected fraction of the elements remaining in $\chi(D)$ that it eliminates is a constant. Even though the bits from the nodes are sent in parallel, we think of the server as processing them sequentially in an arbitrary order within each round. Thus, let $I_t$ be the $t$th bit of information sent by the nodes. (In addition to the fingerprint bits, the nodes send other information, to be described below. These are included in the sequential ordering.) Let $\chi_t(D)$ be the elements of $\chi(D)$ that are still possible given the first $t$ bits of information received by the server. We say that fingerprint bit $I_t$, sent by node $j$, is *unbalanced* if there is an $x'_j$ such that for more than half of the elements $X \in \chi_{t-1}(D)$, $x'_j = x_j$. If there is no such a value of $x_j$, then the bit $I_t$ is called *balanced*. Note that the server can decide if a bit $I_t$ is balanced without any knowledge of $\bar{X}$ other than the structure of $\chi_{t-1}(D)$. Furthermore, for a balanced bit $I_t$ sent by node $i$, at most half the elements of $\chi_{t-1}(D)$ can agree with $\bar{X}$ on $x_i$; the other elements of $\chi_{t-1}(D)$ can potentially be eliminated by $I_t$. Thus, the expected fraction of $\chi_{t-1}(D)$ eliminated by $I_t$ is at least a sixth. Our protocol requires the nodes to send the server $\Theta(m)$ balanced bits. The key will be to do so without sending more than $O(m)$ unbalanced bits.

Before we describe how we do so, we point out another difficulty: determining how many bits each node sends during each round. We could construct a protocol where each node sends a single bit in each round, but such a protocol would not be efficient in terms of total rounds. However, sending more bits per round can be inefficient in terms of the total number of bits the nodes send. To see why, let's make the (very optimistic) assumption that each fingerprint bit a node sends is balanced, until some point in time where it is *defined*. We say node $j$ is defined at time $t$ if all of the elements of $\chi_{t-1}(D)$ have the same string for $x_j$. Thus, a node is defined if and only if the server knows the string for that node.

Let $k_i$ be the number of undefined processor remaining at the start of round $i$. Note that $k_1 = k$. A simple protocol would be to have each undefined processor send $m/k_i$ fingerprint bits in round $i$. However, consider what happens if in each round, all but a fraction of $\frac{1}{\log k}$ of the undefined nodes are made defined by (say) a single fingerprint bit sent from a node that remains undefined. If this continues for $\frac{\log k}{2 \log \log k}$ rounds, there are still undefined nodes. However, in each round, the nodes send $m$ bits, and yet the total number of balanced bits received after all those rounds is $o(m)$. Thus, it is possible that this simple protocol sends a factor of $\frac{\log k}{\log \log k}$ more total bits from the nodes than the $O(m)$ we were hoping for. We shall deal with this with a form of "slow start": as nodes become defined, the remaining nodes do not increase the number of bits they send in each round too quickly. This leads to more total rounds, but ensures that the expected total number of bits sent by the nodes is $O(m)$.

### 2.1 Description of support uniform algorithm

Our algorithm consists of a series of phases, each consisting of two rounds. In phase $i$, let $\mathcal{U}_i$ be the set of nodes that are undefined at the start of phase $i$, and again, $k_i = |\mathcal{U}_i|$. In the first round of phase $i$, every node in $\mathcal{U}_i$ sends $b_i = \min(\frac{3}{2}b_{i-1}, \lceil m/k_i \rceil)$ fingerprint bits, each specified by a fingerprint function chosen randomly and sent to the nodes by the server. We define $b_1 = k$. The server then chooses any sequential ordering of the fingerprint bits received in a round, and processes them in that order. When each fingerprint bit is processed, the inputs in $\chi(D)$ that do not agree with that bit are discarded from consideration by the server.

Before processing each fingerprint bit $I_t$ sent by node $j$, the server checks to see if it is an unbalanced bit: i.e., if some string $x'_j$ agrees with more than half of the elements of $\chi_{t-1}(D)$. For each node $j$ in phase $i$, if there is any unbalanced bit, round $i$ is referred to as being unbalanced for node $j$. We let $h_{ij}$ be the string $x'_j$ that agrees with at least half of the inputs in $\chi_{t-1}(D)$, where $I_t$ is the first unbalanced bit sent by node $j$ in phase $i$. $h_{ij}$ is called the *heavy string* of unbalanced round $i$ for node $j$. The server also keeps track of the total number of balanced bits that it has received.

During the second round of each phase $i$, let $t$ be the total number of bits received by the server through the end of the first round of that phase. The server checks each heavy string $h_{ij}$ to see if it is still consistent with some input in $\chi_t(D)$ and if node $j$ is still undefined. If so, it sends the string $h_{ij}$ to node $j$ . Node $j$ responds with a single bit indicating whether or not $h_{ij} = \bar{x_j}$. These queries are refereed to as *heavy queries*; they and their responses are performed in parallel for all nodes that require them.

The server continues this process until it has received $c \cdot m$ balanced bits, for a constant $c$ to be described below, or until there is only one possible input remaining in $\chi(D)$. It then sends a description of the remaining possible inputs to the nodes, sending the appropriate string of the input to each node. Each node $i$ then sends the index of the string within this list that corresponds to $\bar{x}_i$. These queries are called *index queries*. We call this algorithm **Multi-Uniform**. It should be clear that this algorithm returns the correct answer; in the next section we analyze its performance.

## 3 Performance of support uniform algorithm

In order to analyze the performance of **Multi-Uniform**, we start by pointing out a useful fact about how the distribution over the elements of $\chi(D)$ evolves as the server collects information from the nodes. In particular, the following claim demonstrates that as the server receives fingerprint bits and responses to queries concerning heavy strings, some of the inputs in $\chi(D)$ may be eliminated, but the relative probability weight between the remaining inputs will be unchanged. Thus, when $D$ has a uniform probability weight over $\chi(D)$, the distribution over the inputs of $\chi(D)$ that are not eliminated by a sequence of fingerprint bits will be uniform even after conditioning on those received bits.

Recall that $I_t$ refers to either a fingerprint bit, or the response to a heavy query. Let $X^i$ be some input in $\chi(D)$. Let $\mathcal{Z}$ be a random variable denoting all the fingerprint functions chosen by the server. Let $D_t(X^i) = \Pr[\bar{X} = X^i | I_1 \ldots I_t, \mathcal{Z}]$, and thus $D_t$ is the distribution $D$ conditioned on all information learned by the server up to bit $I_t$.

**Claim 1** *For all $X^i$, $X^j$, and $t$, if $D_t(X^i) > 0$ and $D_t(X^j) > 0$, then $\frac{D_t(X^i)}{D_t(X^j)} = \frac{D(X^i)}{D(X^j)}$.*

*Proof:* From the definition of conditional expectation, we see that

$$D_t(X^i) = \Pr[\bar{X} = X^j | I_1 \dots I_t, \mathcal{Z}] = \frac{\Pr[(\bar{X} = X^i \cap I_1 \dots I_t) | \mathcal{Z}]}{\Pr[I_1 \dots I_t | \mathcal{Z}]}.$$

However, $\mathcal{Z}$ and $\bar{X}$ together specify all information sent by the nodes. Thus, either $\Pr[(\bar{X} = X^i \cap I_1 \dots I_t) | \mathcal{Z}] = \Pr[\bar{X} = X^i | \mathcal{Z}]$ or $\Pr[(\bar{X} = X^i \cap I_1 \dots I_t) | \mathcal{Z}] = 0$, depending on whether or not $X^i$ leads to bits $I_1 \dots I_t$ being sent on $\mathcal{Z}$. However, we assume that $D_t(X^i) > 0$, and so $Pr[(\bar{X} = X^i \cap I_1 \dots I_t) | \mathcal{Z}] = Pr[\bar{X} = X^i | \mathcal{Z}] = Pr[\bar{X} = X^i] = D(X^i)$. Thus, $D_t(X^i) = \frac{D(X^i)}{\Pr[I_1 \dots I_t | \mathcal{Z}]}$, and similarly $D_t(X^j) = \frac{D(X^j)}{\Pr[I_1 \dots I_t | \mathcal{Z}]}$. The claim follows. ■

This leads us to the following central lemma:

**Lemma 1** *The expected number of fingerprint bits sent by the nodes during* **Multi-Uniform** *is* $O(m + k)$.

*Proof:* We demonstrate that the expected number of unbalanced bits that are sent is $O(m + k)$. To do so, we use an accounting scheme where we charge each unbalanced bit sent after the first round to either a balanced bit, or to a bit that is sent in the first round. We do so in a way that guarantees that the expected number of times that any bit is charged is a constant.

Each unbalanced bit is charged to a bit sent previously by the same node. In particular, an unbalanced bit sent in round $i > 1$ by node $j$ is charged to a balanced bit sent by $j$ in the latest round $i' < i$ such that round $i'$ is balanced for node $j$. If there is no such round, then the unbalanced bit is charged to a bit sent by $j$ in the first round. The unbalanced bits charged to bits sent by node $j$ in any round $i'$ are distributed as evenly as possible among the bits in round $i'$ sent by $j$: i.e., the difference between the most unbalanced bits charged to one of $j$'s bits in round $i'$ and the least bits charged is no more than 1.

Let $c_t$ be the random variable indicating how many unbalanced bits are charged to bit $I_t$. If $I_t$ is neither a fingerprint bit sent in the first round nor a balanced bit, then $c_t = 0$. Otherwise, $c_t$ can be larger, but the following claim bounds how much larger.

**Claim 2** *For any* $t$, $E[c_t] = O(1)$.

*Proof (of Claim):* Consider some round $i$ that is unbalanced for node $j$. Let $I_u$ be the first unbalanced bit sent by $j$ in round $i$, and let $h_{ij}$ be the corresponding heavy string. Let $F_{ij}$ be the event that node $j$ is defined at the end of the second round of the phase containing round $i$. We first point out that $\Pr[F_{ij} | I_1 \dots I_{u-1}] \geq \frac{1}{2}$. To see this, note that since $I_u$ is unbalanced, $h_{ij}$ corresponds to at least half of the elements in $\chi_u(D)$. By Claim 1, we see that $D_u$ is uniform over the elements of $\chi_u(D)$, and so $\Pr[\bar{x_j} = h_{ij} | I_1 \dots I_{u-1}] \geq \frac{1}{2}$. The second round of the phase will cause node $j$ to be defined whenever it is the case that $\bar{x_j} = h_{ij}$.

A bit sent by node $j$ in the first round of phase $p$ can only be charged to by a bit sent by node $j$ in phase $p + v$, for $v$ a positive integer, if the first round of all phases $p + 1, \dots, p + v$ are unbalanced. Since a defined node no longer sends any fingerprint bits, the probability of that happening is at most $\frac{1}{2^{v-1}}$. Since the number of bits sent by node $j$ increases from phase to phase by at most a factor of 3/2, the expected number of bits charged to a bit in phase $p$ can be at most $\sum_{v=1}^{\infty} (\frac{1}{2})^{v-1} (\frac{3}{2})^v = O(1)$. ■

Since the number of bits sent in the first round is $k\lceil\frac{m}{k}\rceil$, and the total number of balanced bits sent in subsequent rounds is at most $cm$, the lemma now follows from the linearity of expectation. ■

**Lemma 2** *Let $T$ be the largest $t$ such that $I_t$ is either a fingerprint bit or a response to a heavy query. $E[|\chi_T(D)|] = O(1)$.*

In other words, in protocol **Multi-Uniform**, the expected number of inputs in $\chi(D)$ that are still possible after receiving all of the fingerprint bits and responses to heavy queries is $O(1)$.

*Proof:* Consider some possible input $X' \in \chi(D)$. When $\bar{X} = X'$, then $X' \in \chi_T(D)$. We demonstrate that when $\bar{X} \neq X'$, $\Pr[X' \in \chi_T(D)] \leq \frac{1}{2^m}$. To see this, let $I_t$ be any balanced bit sent by node $j$. The fact that $I_t$ is balanced, combined with Claim 1, implies that $\Pr[\bar{x_j} = x'_j | I_1 \ldots I_{t-1}] \leq \frac{1}{2}$. If $\bar{x_j} \neq x'_j$, then the probability that $X'$ is eliminated by $I_t$ is at least $\frac{1}{3}$. Thus, $\Pr[X' \in \chi_t(D) | I_1 \ldots I_{t-1}] \leq \frac{5}{6}$. In particular, this holds even when $I_1 \ldots I_{t-1}$ have not already eliminated $X'$. Therefore, the probability that $X'$ is not eliminated after $cm$ balanced bits, for $c = \log_{6/5} 2$, is at most $\frac{1}{2^m}$. The lemma now follows from the linearity of expectation. ■

**Lemma 3** *The expected number of rounds is $O(1 + \log m)$.*

*Proof:* Call the first round of a phase *sparse* if the total number of fingerprint bits sent in that round is at most $\frac{2}{3}m$. In any sparse round, the number of bits sent by a node that is not yet defined increases by a factor of $\frac{3}{2}$. Since no node sends more than $m$ bits in a round, the total number of sparse rounds can be at most $O(\log m)$. Since the expected number of fingerprint bits that are sent is $O(m)$, the expected number of rounds that are not sparse can be at most $O(1)$. ■

**Theorem 1** *The protocol* **Multi-Uniform** *has the following performance: the total expected number of bits sent by the nodes is $O(m + k)$, the expected number of bits sent by the server is $O(kn + m \log n)$, and the total expected number of rounds is $O(1 + \log m)$.*

*Proof:* The bound on the expected number of rounds was already stated in Lemma 3. For the bound on the expected number of bits sent by the nodes, note that nodes send three types of bits: fingerprint bits, yes/no responses to heavy queries, and bits responding to index queries at the end of the protocol. We saw in Lemma 1 that the expected number of fingerprint bits is $O(m + k)$. From Lemma 2, we see that the total expected number of index bits is a constant per node. We argued in the proof of Claim 2 that whenever a node sends an unbalanced bit, it will be defined by the end of the phase containing that bit with probability at least $\frac{1}{2}$. Therefore, the expected number of bits sent in response to heavy queries is also a constant per node.

As for the expected number of server bits, note that the server must send three different types of bits: descriptions of fingerprint functions, heavy queries, and index queries. The expected number of fingerprint bits is $O(m)$, and each can be described using $O(\log n)$ bits. Thus the expected number of bits of the first type is $O(m \log n)$. Since the expected number of yes/no responses to heavy queries from each node is a constant, the total expected number of server bits of the second type is $O(n)$ per node. Lemma 2 implies that the expected number of bits of the third type is also $O(n)$ per node. Thus, the total expected number of server bits is $O(nk + m \log n)$. ■

# 4 Extension to arbitrary distributions

We now turn our attention to arbitrary probability distributions. We combine the protocol **Multi-Uniform** with a technique used in [2] to obtain a protocol that requires a constant expected number of rounds when $k = 1$. In this technique, we sort the possible inputs from most likely to least likely. Let $\beta^1$ be the first $h_1$ inputs in this sorted order, where $h_1$ is chosen so that $|\sum_{X \in \beta^1} D(X) - \frac{1}{2}|$ is minimized. In other words, $\beta^1$ has as close to half of the total probability weight as possible. $\beta^2$ has as close to half the remaining probability weight as possible, and in general, $\beta^i$ has as close as possible to half the probability weight not included in $\beta^1 \ldots \beta^{i-1}$. For ease of presentation, we here make the assumption that for some $v \leq 2^{kn}$, $\sum_{X \in \beta^i} D(X) = \frac{1}{2^i}$ for all $i < v$, and $\beta^v$ consists of a single input $X$ such that $D(X) = \frac{1}{2^{i-1}}$. The case where the partition is not this exact leads to a number of technical details which we defer to the full version of the paper. However, this simplified case contains all of the major concepts required for the general case.

In our protocol for the case of arbitrary distributions, which we call **Multi-Arbitrary**, the server tests each of the sets $\beta^i$, in order starting with $\beta^1$, to see if $\bar{X} \in \beta^i$ until it finds the correct such set. This test is done using a variant of the protocol **Multi-Uniform**, and when it reaches the correct set $\beta^i$, it provides the server with the identity of the input $\bar{X}$. It is easy to see that the expected number of sets $\beta^i$ that need to be examined is a constant. Furthermore, techniques from [2] are sufficient to show that if the expected number of bits sent by the nodes for the set $\beta^i$ is $O(\log h_i)$ (which it will be), then the expected total number of bits sent by the nodes is $O(H(D))$.

To use **Multi-Uniform** to test if $\bar{X} \in \beta^i$, the server basically makes the assumption that the probability distribution is support uniform, with the support set $\beta^i$. Thus, $\beta^i$ will be used as the set $\chi(D)$ is used in **Multi-Uniform**. However, there are two problems to overcome: first, the probability distribution over the strings in $\beta^i$ is no longer uniform. Second, in the support uniform case, we are guaranteed that $\bar{X} \in \chi(D)$, whereas in the case of **Multi-Arbitrary**, there is some probability that $\bar{X}$ is not in the current set $\beta^i$. While this probability starts out at approximately $1/2$ for each set $\beta^i$ considered, it can become quite large as elements of $\beta^i$ are eliminated. The technique used to collect balanced bits can cause the total probability weight of the remaining elements of $\beta^i$ to be much smaller than the total probability weight of the remaining elements of the other sets.

In what follows, we demonstrate that these problems can be overcome by an appropriate generalization of the definition of balanced and unbalanced bits for the case of arbitrary probability distributions. In particular, we say that a fingerprint bit $I_t$ sent by node $j$ is *balanced* if there is no value $x'_j$ such that $\Pr[\bar{x_j} = x'_j | I_1 \ldots I_t] > \frac{1}{2}$, where $I_1 \ldots I_t$ represents (as in the protocol **Multi-Uniform**) all of the information learned by the server thus far. In addition to accounting for a non-uniform probability distribution over the elements of $\beta^i$, a key property of this generalized definition of balanced bits that our protocol exploits is that it is with respect to the probability distribution over all of the remaining inputs, and not just those in the current set $\beta^i$ being tested.

The protocol proceeds as follows: the server tests each set $\beta^i$ in order, starting with $\beta^1$. For each such test, the server uses the protocol **Multi-Uniform** with the following changes, where we refer to the call to this protocol used to process $\beta^i$ as **Multi-Uniform**$^i$:

- The value of $m$ used during **Multi-Uniform**$^i$ is $\lceil \log h_i \rceil$.
- During **Multi-Uniform**$^i$, the probability distribution $D$ is conditioned on the information learned in previous calls. Thus, elements of $\beta^i$ may be eliminated even before **Multi-**

**Uniform**$^i$ starts.

- A node $j$ is considered to be defined during **Multi-Uniform**$^i$ if there is a string $x'_j$ such that $\forall X \in \beta^i$ that are still possible given the information received thus far, $x_j = x'_j$.
- The server uses the generalized definition of balanced and unbalanced bits described above. Note that this may mean that during **Multi-Uniform**$^i$, the heavy string for an unbalanced bit may not be consistent with any $X \in \beta^i$.
- When the server sends the index queries to the nodes at the end of **Multi-Uniform**$^i$, each node is allowed to respond "none of the above".

It should be clear that this protocol is guaranteed to return the correct answer. The following theorem describes its efficiency.

**Theorem 2** *Protocol* **Multi-Arbitrary** *has the following performance: the expected total number of bits sent by the nodes is* $O(H(D) + k)$*, the expected number of bits sent by the server is* $O(kn + H(D) \log n)$*, and the expected number of rounds is* $O(1 + \log\lceil H(D)\rceil)$*.*

*Proof:* We first demonstrate that Lemmas analogous to Lemmas 1 and 2 can be proven in this more general scenario. In the following, let $I_t^i$ be the $t$th bit of information sent by the nodes during **Multi-Uniform**$^i$ (again, subject to an arbitrary ordering of the bits sent within the same round), including the responses to the index queries at the end of **Multi-Uniform**$^i$. Let $I(i,t)$ be all bits of information sent before $I_t^i$, including those sent in **Multi-Uniform**$_1$ through **Multi-Uniform**$_{i-1}$. Let $M^i$ be the event that $\bar{X} \notin \beta^{i'}$ for any $i' < i$.

**Lemma 4** *The expected number of fingerprint bits sent by the nodes during* **Multi-Uniform**$^i$*, given* $M^i$*, is* $O(k + \log h_i)$*.*

*Proof:* We use the same charging scheme to bound the number of unbalanced bits as in the proof of Lemma 1, where the "first round" is here considered to be the first round of **Multi-Uniform**$^i$. Using the generalized definition of balanced bits, we see that it is still the case that $\Pr[\bar{x_j} = h^i_{qj} | I(i,u)] \geq \frac{1}{2}$, where $I^i_u$ is the first unbalanced bit sent by node $j$ in round $q$ of **Multi-Uniform**$^i$, and $h_{qj}$ is the corresponding heavy string. Thus, the same argument as in the proof of Lemma 1 demonstrates that this scheme still only charges an expected constant number of bits to each balanced bit or bit sent in the first round of **Multi-Uniform**$^i$. ■

Let $\beta_t^i$ be the elements of $\beta^i$ that are still possible given $I(i,t)$ and $I_t^i$. Let $T^i$ be the largest $t$ such that $I_t^i$ is either a fingerprint bit or a response to a heavy query in **Multi-Uniform**$^i$. Let $s(\beta_t^i)$ be the number of elements in $\beta_t^i$.

**Lemma 5** $E[s(\beta^i_{T^i})|M^i] = O(1)$.

*Proof:* The proof is nearly identical to the proof of 2. Consider any input $X' \in \beta^i$ such that $X' \neq \bar{X}$. For any balanced bit $I_t^i$ sent by some node $j$, the generalized definition of balanced bits implies that $\Pr[\bar{x_j} = x'_j | I(i,t)] \leq \frac{1}{2}$. Thus, the probability that any given balanced bit eliminates $X'$ is at least $\frac{1}{6}$. This implies that $\Pr[X' \in \beta^i_{T_i} | M^i] \leq \frac{1}{h_i}$. Even though some elements of $\beta^i$ may

be eliminated before **Multi-Uniform**$^i$, we still have $s(\beta^i) \leq h_i$. Thus, the Lemma follows by the linearity of expectation. ■

We next use the following claim, which is implied by the argument in Theorem 2 of [2].

**Claim 3** *There is a constant $c$ such that for any distribution $D$, $\sum_i \frac{\log h_i}{2^{i-1}} \leq cH(D)$.*

It is easy to see that the protocol must process the set $X_i$ if and only if $M^i$ holds, and that $\Pr[M^i] = \frac{1}{2^{i-1}}$. Thus, Lemma 4 combined with Claim 3 tells us that the expected number of fingerprint bits sent by the nodes during all of **Multi-Arbitrary** is $O(H(D) + k)$. Furthermore, from Lemma 5, the expected number of index bits sent during **Multi-Uniform**$^i$, given $M^i$, is a constant per node. Thus, the expected total number of index bits sent during all of **Multi-Arbitrary** is also a constant per node. Similarly, as in the proof of Theorem 1, we can argue that the expected number of responses to heavy queries is also a constant per node.

To see that the expected number of bits sent by the server is $O(kn + H(D) \log n)$, note that Claim 3 implies that it suffices to show that the expected number of server bits sent during **Multi-Uniform**$^i$, given $M^i$, is $O(kn + (\log h_i)(\log n))$. This now follows from an argument analogous to that used to bound the expected number of server bits in Theorem 1. Thus, the theorem now follows from the following Lemma:

**Lemma 6** *The expected number of rounds required by protocol* **Multi-Arbitrary** *is* $O(1+\log\lceil H(D)\rceil)$.

*Proof:* Using the same argument as in the proof of Lemma 3, we see that there is a constant $c_1$ such that the expected number of rounds required by **Multi-Uniform**$^i$, given $M^i$, is at most $c_1(1 + \log\log h_i)$. Thus, the expected number of rounds used by the protocol is at most $c_1 \sum_i \frac{1+\log\log h_i}{2^{i-1}}$. Claim 3 implies that each individual term of the sum $\sum_i \frac{\log h_i}{2^{i-1}}$ must be at most $cH(D)$, and thus we see that $\forall i, \log h_i \leq c2^{i-1}H(D)$. Thus, the expected number of rounds required is at most $c_1 \sum_i \frac{1+\log(c2^{i-1}H(D))}{2^{i-1}} \leq 2c_1(1 + \log(cH(D))) + \sum_i \frac{i}{2^i} = O(1 + \log H(D))$. ■ ■

## 5 Reducing the number of server bits by allowing errors.

We next point out that if we allow the protocol to make errors, then the number of server bits can be reduced. In order to do so, we introduce the protocol **Multi-Error**, which we now describe.

During the protocol **Multi-Uniform**, a significant portion of the bits sent by the server are used to describe various $n$-bit strings during heavy queries and index queries. These strings are sent to some node $j$ to perform an equality test: "Is $\bar{x_j} = x'_j$?" The protocol **Multi-Error** mimics **Multi-Arbitrary**, except that in each subroutine **Multi-Uniform**$^i$, these equality tests are performed using a standard fingerprinting technique for such tests. In particular, for a heavy query, instead of sending the $n$-bit string $h_{ij}$ to node $j$, the server chooses a random prime $p$ of size at most $(\frac{nk}{\Delta})^2$, and sends node $j$ this prime as well as the value $int(h_{ij}) \bmod p$, where $int(h_{ij})$ is the $n$-bit string $h_{ij}$ interpreted as an integer between 0 and $2^n - 1$. The node $j$ checks to see if $int(h_{ij}) = int(\bar{x_j}) \bmod p$, and sends a single bit response indicating the result of this test back to the server. In the case of equality, the server assumes for the remainder of the protocol that the strings are equal. Note that information received later by the node or by the server may contradict this. In that case, **Multi-Error** gives up and declares an error.

When the server sends the nodes index queries at the end of **Multi-Uniform**$^i$, this is treated similarly to heavy queries. The only difference is that the server may now have $r$ different strings $x_j^1, \ldots, x_j^r$ that it would send to node $j$, instead of just the one string $h_{ij}$. Thus, the protocol first performs a pre-processing step to narrow the set of possibilities down to one. To do so, the server chooses a random prime $p$ of size at most $n^2r^3$. It then checks to see if $\exists u \neq v$ such that $int(x_j^u) = int(x_j^v) \bmod p$. If so, it chooses a new random prime $p$, until no such $u$ and $v$ exist.[3] The server then sends node $j$ the prime $p$, as well as $int(x_j^u) \bmod p$, for $1 \leq u \leq r$. If node $j$ sees that $int(x_j^u) = int(\bar{x_j}) \bmod p$ for some $u$, it sends the server the index $u$. The server and node then perform the same procedure with $x_j^u$ as in the case of a heavy query. Otherwise, the node $j$ informs the server that $\bar{X} \notin \beta^i$, and the protocol continues on to **Multi-Uniform**$^{i+1}$.

The definition of balanced and unbalanced bits in the protocol **Multi-Error** is the same as in **Multi-Arbitrary**, in the sense that it takes into account all the information learned by the server thus far. Note, however, that the information learned during an equality test is now slightly different; the server takes this into account when determining whether or not a bit is balanced. In particular, if a node indicates during an equality test that the fingerprints are different, then in addition to the string being tested, the server is also able to eliminate any other string that would have led to the same fingerprint. In the case where the node indicates that the fingerprints are the same, again, the server assumes that the underlying strings really are the same, and computes whether or not bits are balanced based on this assumption.

This assumption does not lead to any problems if we condition on there being no false positives (i.e., equality tests where the fingerprints are the same but the underlying strings are different). In the case where there is a false positive, the server has an incorrect view of the conditioned probability distribution over inputs, and thus may incorrectly classify subsequent bits as balanced and unbalanced. This makes the performance of the resulting protocol execution difficult to predict. To deal with this, we impose an upper bound on the total number of bits the server sends and receives in the protocol **Multi-Error**. This, combined with a small probability of ever encountering a false positive, allows us to provide an upper bound on the expected number of bits sent and received by the server in **Multi-Error**.

In more detail, in **Multi-Error**, the server keeps track of the total number of bits it has sent and received. If the number it sends ever exceeds $n[k \log(\frac{kn}{\Delta}) + H(D) \log n]$, then the server informs the nodes of this fact, and the protocol switches to the protocol **Multi-Arbitrary**, where the server discards all information it has received thus far. We refer to this as *falling back on* **Multi-Arbitrary**. If the number of bits it receives ever exceeds $n[H(D) + k]$, then it informs the nodes of this fact, and the nodes then simply send their strings directly to the server. We refer to this as *falling back on uncompressed transmission.*

**Theorem 3** *Given any $\Delta > 0$,* **Multi-Error** *allows the server to determine an input $X'$ such that $Pr[X' = \bar{X}] \geq 1 - \Delta$, and the expected total number of bits sent by the nodes is $O(H(D) + k)$, the expected number of bits sent by the server is $O(k \log(\frac{kn}{\Delta}) + H(D) \log n)$, and the expected number of rounds is $O(1 + \log\lceil H(D)\rceil)$.*

*Proof:* For a run of the protocol where there is no false equality, **Multi-Error** will return the correct answer. In this case, the bounds on the performance of the protocol follow from the same arguments as for **Multi-Uniform**, except that we can replace the $n$-bit strings sent by the server

[3] Note that this is not unduly expensive computationally for the server: the probability that a choice of $p$ is suitable is at least $\frac{1}{2}$.

with the description of the equality fingerprint we use, which requires only $O(\log(\frac{kn}{\Delta}))$ bits. The only other difference is that if the number of bits sent by the server or by the nodes becomes too large, then the protocol falls back on a different tactic. However, by Markov's inequality, the probability that the number of server bits used in **Multi-Error** exceeds $n[k\log(\frac{kn}{\Delta}) + H(D)\log n]$ is $O(\frac{1}{n})$. Thus, the contribution of falling back on **Multi-Arbitrary** to the expected number of bits sent by the server is $O(k + \frac{H(D)\log n}{n})$. Similarly, the contribution to the expected number of bits sent by the nodes of falling back on uncompressed transmission is $O(k)$. Thus, to analyze the protocol, we only need to demonstrate that the probability of a false positive is not too large, and that the contribution of runs of the protocol containing false positives to the total expected number of bits sent by the server or by the nodes is not too large.

We next point out that the probability that the protocol ever makes an error is at most $\Delta$. Standard arguments concerning the equality test we use demonstrate that the probability of a false positive on any given test is at most $\frac{\min(1/n,\Delta)}{8k}$. Let $B^i_{tj}$ be the event that the first false positive of **Multi-Error** is during the $t$th heavy query performed with node $j$ during **Multi-Uniform**$^i$ (and hence the protocol does in fact reach **Multi-Uniform**$^i$, and performs at least $t$ heavy queries with node $j$ within **Multi-Uniform**$^i$). The probability of reaching **Multi-Uniform**$^i$, assuming that there have been no prior false positives, is at most $\frac{1}{2^{i-1}}$. Similarly, the probability of reaching the $t$th heavy query within **Multi-Uniform**$^i$, assuming no prior false positives and that the protocol reaches **Multi-Uniform**$^i$, is at most $\frac{1}{2^{t-1}}$. Thus, $\Pr[B^i_{tj}] \leq \frac{\min(1/n,\Delta)}{k2^{i+j+1}}$. Similarly, the probability of the first false positive occurring during the index query for node $j$ during **Multi-Uniform**$^i$ is at most $\frac{\min(1/n,\Delta)}{k2^{i+1}}$. Therefore, by using a union bound over all nodes and all possible times for each node to have the first false positive, we see that the probability of any false positive is at most $\min(1/n, \Delta)$.

It only remains to bound the contribution of runs of **Multi-Error** containing false positives to the total expected performance of the protocol. However, we have constructed **Multi Error** in such a way that the maximum number of server bits that can ever be sent is $n[k\log(\frac{kn}{\Delta}) + H(D)\log n]$, not counting falling back on **Multi-Arbitrary**. Since the probability of a false positive is at most $\frac{1}{n}$, the contribution of runs of the protocol containing false positives to the expected number of server bits sent is $O(k\log(\frac{kn}{\Delta}) + H(D)\log n])$. Similarly, the contribution of false positives to the expected number of bits sent by the nodes is $O(H(D) + k)$. ■

# References


[1] A. Aaron and B. Girod, "Compression with side information using turbo codes," Proc. DCC'02, Snowbird, UT, April 2002.

[2] M. Adler and B. Maggs. Protocols for asymmetric communication channels. *Journal of Computer and System Sciences* 63(4), pages 573–596, December 2001. (Special issue of best papers from FOCS 1998.)

[3] J. Bajcsy and P. Mitran, "Coding for the Slepian-Wolf problem with turbo codes," Proc. GlobeCom'01, San Antonio, TX, November 2001.

[4] P. Bose, D. Krizanc, S. Langerman, and P. Morin. Asymmetric communication protocols via hotlink assignments. *Theory of Computing Systems*, 36(6):655-661, 2003. Special issue of selected papers from *The IXth International Colloquium on Structural Information and Communication Complexity* (SIROCCO 2002).

[5] J. Chou, D. Petrovic, and K. Ramchandran, "A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consuumption in Sensor Networks," In Proc. of Infocom 2003.

[6] T. Coleman, A. Lee, M. Medard, and M. Effros, "On some new approaches to practical Slepian-Wolf compression inspired by channel coding," Proc. DCC'04, Snowbird, UT, March 2004.

[7] R. Cristescu, B. Beferull-Lorazano, and M. Vetterli, "On Network Correlated Data Gathering," in Proc. of Infocom 2004.

[8] Nick Feamster, Magdalena Balazinska, Greg Harfst, Hari Balakrishnan, and David Karger, "Infranet: Circumventing Censorship and Surveillance," In Proc. 11th USENIX Security Symposium, San Francisco, CA, August 2002. (Awarded Best Student Paper.)

[9] L. M. Feeney and M. Nilsson "Investigating the energy consumption of a wireless network interface in an ad hoc network", In INFOCOM, 2001.

[10] J. Garcia-Frias and Y. Zhao, Compression of correlated binary sources using turbo codes, IEEE Communications Letters, vol. 5, pp. 417-419, October 2001.

[11] J. Garcia-Frias and W. Zhong, "LDPC codes for compression of multiterminal sources with hidden Markov correlation," IEEE Communications Letters, pp. 115-117, March 2003.

[12] S. Ghazizadeh, M. Ghodsi, and A. Saberi, "A New Protocol for Asymmetric Communication Channels, Reaching Lower Bounds," *Scientia Iranica*, 8(4), 2001.

[13] Leonardo Holanda and Eduardo Laber, "Improved bounds for asymmetric communication protocols," *Information Processing Letters* 83(4), pp 205-209.

[14] C. Lan, A. Liveris, K. Narayanan, Z. Xiong, and C. Georghiades, "Slepian-Wolf coding of multiple M-ary sources using LDPC codes," Proc. DCC'04, Snowbird, UT, March 2004.

[15] A. Liveris, Z. Xiong, and C. Georghiades, "Distributed compression of binary sources using conventional parallel and serial concatenated convolutional codes," Proc. DCC'03, Snowbird, UT, March 2003.

[16] A. Liveris, Z. Xiong and C. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," IEEE Communications Letters, vol. 6, pp. 440-442, October 2002.

[17] A. Liveris, C. Lan, K. Narayanan, Z. Xiong, and C. Georghiades, "Slepian-Wolf coding of three binary sources using LDPC codes," Proc. Intl. Symp. Turbo Codes and Related Topics, Brest, France, September 2003.

[18] D. Marco, E. Duarte-Melo, M. Liu, D. Neuhoff, "On the Many-To-One Transport Capacity of a Dense Wireless Sensor Network and the Compressibility of Its Data," in Proc. International Workshop on Information Processing in Sensor Networks (IPSN), April 2003.

[19] S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," IEEE Signal Processing Magazine, vol. 19, pp. 51-60, March 2002.

[20] D. Schonberg, S. Pradhan, and K. Ramchandran, "Distributed code constructions for the entire Slepian-Wolf rate region for arbitrarily correlated sources," Proc. DCC'04, Snowbird, UT, March 2004.

[21] D. Slepian and J.K. Wolf. Noiseless encodings of correlated information sources. *IEEE Trans. on Information Theory*, IT-19:471–480, July 1973.

[22] A. Scaglione and S. Servetto, "On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks," in Proc. of Mobicom 2002.

[23] V. Stankovic, A. Liveris, Z. Xiong, and C. Georghiades, "Design of Slepian-Wolf codes by channel code partitioning," Proc. DCC'04, Snowbird, UT, March 2004.

[24] John Watkinson, "New Protocols for Asymmetric Communication Channels," Masters Thesis, university of Toronto, 2000.

[25] John Watkinson, Micah Adler, and Faith Fich, "New Protocols for Asymmetric Communication Channels", in *Proceedings of 8th International Colloquium on Structural Information and Communication Complexity* (Sirocco) 2001.

[26] Zixiang Xiong, Angelos D. Liveris, and Samuel Cheng, "Distributed Source Coding for Sensor Networks," to appear in IEEE Signal Processing Magazine, September 2004. Web version available at `http://lena.tamu.edu/~zx/papers/SPM.pdf`.

[27] A.C. Yao, "Some Complexity Questions Related to Distributive Computing," In *Proc. of 11th ACM Symposium on Theory of Computing*, (STOC) pp. 209-213, 1979.