Joint Identity-Message Coding

L. Keller, M. Jafari Siavoshani, C. Fragouli, K. Argyraki, and S. Diggavi

Abstract—**In a significant class of sensor-network applications, the identities of the reporting sensors constitute the bulk of the communicated data, whereas the message itself can be as small as a single bit—for instance, in many cases, sensors are used to detect whether and where a certain interesting condition occurred, or to track incremental environmental changes at fixed locations. In such scenarios, the traditional network-protocol paradigm of separately specifying the source identity and the message in distinct fields leads to inefficient communication.**

This work addresses the question of how communication should happen in such identity-aware sensor networks. We calculate theoretical performance bounds for this type of communication, where "performance" refers to the number of transmitted bits. We propose a communication protocol, where the identity and message of each source are specified jointly using subspace coding. We show through analysis and simulation that our protocol's performance is close to optimal and compare it to the performance of a traditional protocol, where identity and message are specified separately.

Index Terms—**Wireless sensor networks, Identity-aware sensors networks, Network coding, Energy efficiency**

I. Introduction

In traditional network protocols, each packet carries its source identity in a dedicated header field, separately from the communicated message, which constitutes the packet's payload. To increase their information rate, several protocols use encoding or compression techniques that look to minimize the size of the message; to the best of our knowledge, none of these techniques consider the source identity as part of the data that needs to be encoded or compressed—and for good reasons: In the typical communication scenarios where encoding or compression makes sense, the message constitutes the bulk of the communicated data, whereas the source-identity overhead is relatively insignificant.

The situation is reversed in wireless sensor networks that monitor the evolution of an environmental variable over time and space: Sensors are often used to track *whether* and *where* a certain condition occurs, *e.g.*, temperature exceeds a threshold, a perimeter is violated, soil or water is contaminated; in other cases, they are used to track incremental changes at fixed locations, *e.g.*, the evolution of snow height at mountain peaks for avalanche prediction or seismic activity for earthquake prediction. In such scenarios, it makes sense to associate each sensor with a fixed location and have it report, periodically, its identity and measurement to a collecting sink; assuming a network of tens or hundreds of nodes, the identities of the reporting nodes now become the bulk of the communicated data, whereas the message itself (*i.e.*, each reported measurement) can be as small as a single bit.

We use the term *identity-aware sensor network* to describe such a paradigm (where each sensor periodically communicates to the sink its identity plus a small message). This is in contrast to the identity-*un*aware sensor networks usually encountered in the literature, where the sink collects functions of the histogram of the sensor measurements, such as the average or maximum temperature measured by all the sensors.

In an identity-aware sensor network, every piece of information (identity and message) produced by each sensor must reach the sink. This affects energy consumption in two ways. First, there is no room for reducing the amount of transmitted traffic (and, hence, overall energy consumption) by combining multiple messages into one value (*e.g.*, their sum), as typically proposed for identity-unaware networks. Second, the "busiest" nodes in the network (*i.e.*, those that are located close to the sink and act as relays for the rest of the network) have to forward significantly more traffic than nodes in the network periphery. For sensor networks operating with renewable energy [1], this raises the amount of per-sensor energy required to run the network for a given amount of time, making it harder to charge the network through natural resources.

With this work, we address the question of how communication should happen in identity-aware sensor networks. Our main observation is that sensor identities are a special kind of data: for a fixed node, the identity is a constant number that does not change with every transmission, in contrast to the messages that do. Thus, we expect to develop a more efficient communication protocol by treating identities specially rather than treating them like messages, and we show that this is indeed the case.

Our contribution is a new, easy-to-implement communication protocol, where the identity and message of each source are jointly specified using subspace coding. To evaluate our protocol, we compute theoretical performance bounds for identity-aware communication, where we consider two performance metrics: *average transmissions* capture the average number of bits that are transmitted by any node in the network, while *busiest-node transmissions* capture the number of bits that are transmitted by the "busiest" node, *i.e.*, the one that has to transmit the most bits. We show that the performance of our protocol is close to the optimal bounds. We also compare it to the performance of a traditional protocol, where the identity and message of each source are specified in separate fields. We show that, while the traditional protocol performs slightly better in terms of average transmissions, our protocol performs significantly better in terms of busiest-node transmissions, and

Manuscript received May 18, 2009; revised February 18, 2010.

The authors are with the School of Computer and Communication Sciences, EPFL, Switzerland

This work was supported by SNF Award No PP002110483, the Hasler Foundation ManCom Project No 2072 and the EU Project N-CRAVE FP7 ICT-2007215252.

The work of Suhas Diggavi was supported in part by the Swiss National Science Foundation NCCR-MICS center.

this advantage increases with the number of nodes in the network.

The basic idea behind our protocol is to assign a different codebook to each sensor and let each sensor implicitly convey its identity through its choice of codebook. We realize this using subspace encoding: each reporting sensor communicates its identity by generating a set of vectors that represent a distinct subspace—distinct from the subspaces generated by all other sensors. By combining incoming vectors, intermediate nodes essentially produce different (compact) representations of the subspaces generated by the reporting sensors. We exploit the invariance properties of subspaces, such that neither sensors nor sink need any knowledge of either network topology or intermediate-node operations in order to generate their vectors or (in the sink's case) decode them.

The rest of the paper is organized as follows. Section II presents examples of identity-aware sensor networks and illustrates the main idea behind our protocol. Section III describes our protocol in detail. Section IV presents our theoretical performance analysis, while Section V presents experimental results obtained through the TOSSIM simulator [2]. Section VI discusses related work, and Section VII concludes the paper.

II. Motivation and Setup

We consider sensor networks where each node communicates its identity and a small (relatively to the identity) message to a collecting sink. This is different from the typical scenarios discussed and analyzed in the literature, where sensor networks are used to compute aggregate statistics (*e.g.*, the average temperature in a building) that do not require associating each message with a specific sensor. Given our departure from the commonly used paradigm, to motivate our work, we first discuss a few applications where sensor identity forms the bulk of the communicated data (Section II-A). Then we illustrate the main idea behind our protocol (Section II-B) and introduce our network model (Section II-C).

A. Applications

In all the applications we consider, the sensors are used to periodically reconstruct the *spatial field* of a physical quantity, *i.e.*, the variation of that quantity as a function of space [3].

Differential Updates: In many cases of environmental monitoring, to avoid unpleasant surprises, we need to choose the measurement frequency such that the spatial field under measurement changes relatively slowly. For instance, when monitoring the level of snow on a mountain surface for avalanche prediction, it makes sense to measure frequently enough, such that, at any location, the snow level never changes by more than 2-4 centimeters between measurements. In such scenarios, each sensor needs to communicate only the difference of its new measurement from the last one, together with its identity. Assuming networks of tens or hundreds of nodes, the identity may require one or two bytes, while the update itself needs only a few bits [3]. Many such examples also arise in security applications, such as the indoor deployment of a sensor network in an office building, to measure atmosphere contamination.

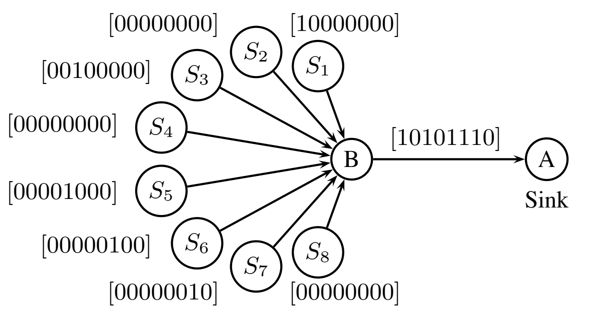


Fig. 1. Sources $S_1, \dots, S_8$ send their identity and a 1-bit message to the sink A through a relay node B.

Spatial Correlation: In other cases, the spatial field under measurement *at a given time* varies smoothly over space—this is the case, for instance, with temperature and pressure [3], [4]. We can leverage such smooth variation by having a set of densely deployed sensors communicate only few bits of information, then use techniques like distributed source coding [5] to reconstruct the entire spatial field. This idea takes advantage of "oversampling" of the sensor field, and collecting coarse information from each sensor.

Multi-stage Collection: Sometimes we are interested in reconstructing only a few "interesting" regions, *e.g.*, examine only the areas where the measurements suggest that there is potential for an avalanche. In such cases, it makes sense to collect data in stages: in the first stage, each sensor communicates its identity along with few bits of information (just enough to get a coarse representation of the field); if something interesting is revealed, the sink queries the relevant sensors for more information in the second stage [6]. In many practical scenarios, it is enough for each sensor to send a single bit of information during the first stage—signaling, for instance, whether a threshold was reached, a perimeter was violated, or an animal was sighted.

B. Basic Idea: Representation of Identities

In the context of the applications discussed above, the traditional approach of keeping the source identity and the message in separate fields leads to inefficient communication. We now illustrate this inefficiency with a simple example and introduce the idea of joint identity-message coding.

Consider the network of Figure 1, where 8 source nodes need to communicate information to a sink node A via a relay node B. Suppose each source $S_i, i = 1, \dots, 8$, needs to communicate its identity and a 1-bit message.

Consider the following communication protocol, which we will call "aggregation": Each source creates a packet that contains 3 bits specifying its identity and 1 bit specifying its message, and sends this to the intermediate node B. To relay this information to A, B could naively forward the 8 packets, which would result in 8 packet transmissions over link BA; to avoid this overhead, B combines the 8×4 identity and message bits into a single packet. This communication protocol can be easily generalized to work on an arbitrary tree: each source sends out a packet with its identity and message specified in separate fields; intermediate nodes aggregate and forward incoming packets toward the sink.

Aggregation results in unequal transmission load over the different network links, with the heavier burden placed on the links closer to the sink. In our example, it results in 4 bits of information being transmitted over each link S_iB as opposed to 32 bits over link BA.

Now consider the following alternative communication protocol, which we will call "coding": Each source S_i sends out an 8-bit packet with its message encoded in bit i and all other bits set to 0; this is the simplest example of using a "code" to represent the identity of a node along with its message. Node B just XORs all incoming packets and sends the resulting 8-bit packet to A, as depicted in Figure 1. Node A can interpret its received message with the understanding that position i corresponds to the message sent by node S_i. Again, this protocol can be easily generalized to work on an arbitrary tree: source S_i sends out a packet with its message specified at bit i and all other bits set to 0; each intermediate node XORs all incoming packets and forwards the one resulting packet toward the sink.

With coding, each node forwards the *same* number of bits, *i.e.*, communication overhead is evenly distributed across the network, alleviating the problem of depleting the battery of the nodes located close to the sink. As a result, compared to aggregation, coding leads to more efficient communication on link BA (8 bits instead of 32); the price we pay is a small decrease in efficiency on the S_iB links, which now have to carry 8 (rather than 4) bits of data. Note that, with coding, node B is not required to understand and process the contents of incoming packets; information is always encoded at its source and decoded at the sink, while each node is oblivious to the codes used by other nodes.

More formally, we propose that each source employs a different *codebook*, *i.e.*, a different mapping of messages to packets; the sink knows the codebook used by each source and, hence, can determine who sent what, *i.e.*, the sender implicitly communicates its identity through its choice of codebook. This approach agrees with the insight we have from information theory: the scenario of Figure 1 is reminiscent of the classic multiple-access channel problem, where multiple users simultaneously transmit to a single receiver over a common channel; it is well known that the users do not have to explicitly specify their identities, as long as they choose distinct enough codebooks that can be disambiguated at the receiver ([7], Chapter 14).

The simple coding we described here is a special case of joint identity-message coding, which is fully described in Section III.

C. Network Model

We consider a network of N sensor nodes and a sink, which operates in rounds. In each round, some or all of the nodes act as "active sources." When node i acts as an active source, it generates exactly one tuple t_i, which has form $\langle i, m_i \rangle$, where i is the identity of the node and m_i is a message that takes values from a set $\mathcal{M}_i$. For simplicity, we assume that all the sets $\mathcal{M}_i$ have the same cardinality, namely $M \triangleq |\mathcal{M}_i| = M, \forall i$.

We assume that the nodes build and maintain a routing tree[1] rooted at the sink, *e.g.*, by using the Collection Tree Protocol (CTP) [8]. The tree does not need to be static, but may adapt dynamically to network conditions. So, when we say that a node "forwards a packet toward the sink," we mean that it sends the packet to its parent on the routing tree.

During each round, each node i that acts as an active source maps its tuple t_i to one or more packets (according to the communication protocol used) and forwards these toward the sink. Each node that has children acts as a relay: it waits to receive all the packets sent by its children, processes them (according to the communication protocol used), and generates new packets to forward toward the sink. So, the set of packets sent out by each node contains all the tuples generated by the node and the node's descendants. We will refer to this set of tuples as the "tuple aggregate" A observed by the node, and we will denote by $I(A)$ the set of sources whose tuples are contained in tuple aggregate A.

As a communication protocol, the network uses either *aggregation* or *coding*.

1) The Aggregation Protocol: Each tuple t_i is mapped to $\lceil \log_2 N \rceil + \lceil \log_2 M \rceil$ bits. The first $\lceil \log_2 N \rceil$ bits encode the identity of the source, while the remaining $\lceil \log_2 M \rceil$ bits encode the message. During each round, each node sends exactly one packet: A node that has no children sends a packet that includes its tuple. A node with children waits to receive all the tuples sent by its children, concatenates them into a single packet together with its own tuple (if it also acted as an active source in the current round), and forwards the resulting packet toward the sink. This means that the more descendants a node has, the more tuples—hence, the more bits—it has to transmit. More specifically, a node that observes a tuple aggregate A containing tuples from a set of sources $I(A)$ has to transmit $|I(A)|(\lceil \log_2 N \rceil + \lceil \log_2 M \rceil)$ bits.

2) The Coding Protocol: Each tuple t_i is mapped to d vectors of length ℓ. During each round, each node sends exactly d vectors of length ℓ: A node that has no children sends the d vectors that represent its tuple. A node with children waits to receive all the vectors sent by its children, linearly combines them with its own d vectors (if it also acted as an active source in the current round) to produce d new vectors, which it forwards toward the sink. This means that, no matter how many descendants a node has, it always has to transmit the same number of bits.

To reason about the coding protocol, we use the notion of *subspace*: A vector of size ℓ belongs to the ℓ-dimensional vector space $V = \mathbb{F}_q^\ell$. A set of d linearly independent vectors of size ℓ spans a d-dimensional subspace of V, denoted by π. We say that two subspaces are *distinct* if they differ in at least one dimension. For instance, each vector sent by source S_i in Figure 1 belongs to the 8-dimensional vector space $\mathbb{F}_2^8$ and represents a 1-dimensional subspace of $\mathbb{F}_2^8$. We will use two subspace operations:

- $\pi_\alpha + \pi_\beta = \{x + y \mid x \in \pi_\alpha, y \in \pi_\beta\}$ is the smallest subspace that contains both π_α and π_β;

[1]In this paper, we focus on routing over trees, however, our protocol can be adapted to other routing structures as well.

- $\pi_\alpha \cap \pi_\beta = \{x \mid x \in \pi_\alpha \text{ and } x \in \pi_\beta\}$ is the largest subspace that belongs to both π_α and π_β.

Moreover, to express how "far apart" two subspaces are we use the *distance* between two subspaces π_α and π_β, defined as

$$d(\pi_\alpha, \pi_\beta) \triangleq \dim(\pi_\alpha + \pi_\beta) - \dim(\pi_\alpha \cap \pi_\beta). \tag{1}$$

So, we can say that, in the coding protocol, each node i uses a codebook $\mathcal{C}_i$, which maps each message to a d-dimensional subspace of $\mathbb{F}_q^\ell$:

$$\begin{aligned}\mathcal{C}_i :& \mathcal{M}_i \to \\ & \{\pi_j^{(i)} : \pi_j^{(i)} \subseteq \mathbb{F}_q^\ell,\ \dim(\pi_j^{(i)}) = d,\ 1 \le j \le |\mathcal{M}_i|\}, \\ & i = 1, \dots, N\end{aligned}$$

To communicate message m_i, node i chooses the corresponding subspace π from $\mathcal{C}_i$ and generates d vectors that span π. For instance, source S_1 in Figure 1 needs to communicate one of two messages, m_1 and m_2. S_1's codebook maps m_1 to the subspace spanned by vector [00000000] and m_2 to the subspace spanned by vector [00000001].

The sink determines the transmitted tuples by observing the subspace spanned by the vectors it receives: Assuming node i generates vectors that span subspace π_i, the sink observes vectors from the subspace $\pi_1+\pi_2+\cdots+\pi_N$. More specifically, if the sink has a single child (as in Figure 1), it will receive exactly d vectors from $\pi_1+\cdots+\pi_N$, which can be represented as $\mathbf{Y} = \sum_{i=1}^N \mathbf{G}_i \mathbf{X}_i$, where $\mathbf{Y}$ is a matrix whose rows are the vectors observed at the sink, $\mathbf{X}_i$ is a matrix whose rows are the original vectors generated by node i, and $\mathbf{G}_i$ is a "mixing matrix" that represents the end-to-end transformation incurred by $\mathbf{X}_i$. The sink does not know the mixing matrices (and neither does any other node), hence, it cannot determine $\mathbf{X}_i$ from $\mathbf{Y}$. However, no matter what the values of $\mathbf{G}_i$ are, as long as there is no end-to-end information loss, if the rows of $\mathbf{X}_i$ spanned subspace π_i, then the rows of $\mathbf{Y}$ will necessarily span subspace $\pi_1 + \pi_2 + \cdots + \pi_N$, because subspaces are unaffected by linear operations.

In order for the sink to unambiguously determine the transmitted tuples, every possible combination of transmitted tuples must result in a different subspace. Hence, we define a *code* as a set of N codebooks, each corresponding to a different node, and we say that a code is *identifiable*, if each distinct set of tuples generated by the nodes can be uniquely decoded at the sink. More formally:

*Definition 1 (**Identifiable Code**):* An identifiable code is a set of N codebooks $\mathcal{C}_i = \{\pi_j^{(i)} : 1 \le j \le |\mathcal{M}_i|\}$, $i = 1, \dots, N$, with $\pi_j^{(i)} \subseteq \mathbb{F}_q^\ell$ d-dimensional subspaces, such that every possible combination of subspaces generated by the N sources and linear operations at the intermediate network nodes, results in a *distinct* subspace observed at the sink.

III. Code Design

We start with the observation that, in many practical scenarios, the network topology and/or the application impose a natural limit on the number of original vectors that get combined within the network. Indeed, two vectors originally generated by two different sources get combined only if their paths to the sink overlap. In Figure 1, the sink has a single neighbor (node B), so all the vectors generated by all the sources get combined at that neighbor.

Now consider a network, where the sink has K neighbors, and sources are symmetrically deployed along these K directions—in which case, we can think of Figure 1 as the "branch" of the network corresponding to one of these neighbors. In this case, each packet that reaches the sink contains approximately $\frac{N}{K}$ combined source vectors. Moreover, in many applications (*e.g.*, anomaly-sensing applications), we expect only a small subset of the sensors to report during each round, hence, fewer than N sets of vectors are generated and get combined within the network.

With this in mind, we formulate our code-design problem as follows: Given N potential sources, we assume that each vector that reaches the sink is a linear combination of original vectors from *at most* K sources (and we do not know which K). Note that multiple such vectors might reach the sink, each potentially containing the combination of a *different* subset of K sources. We want to design codes that allow the sink to look at each received vector and determine (i) which is the corresponding subset of sources and (ii) what are their messages. Note that $K = N$ corresponds to the special case where all sources are active and all original vectors may be combined.

For simplicity, we first describe a code for the case where each sensor either communicates a single bit of information (to indicate that a certain event occurred) or remains silent (to indicate that it didn't) (Section III-A). We later generalize to an arbitrary message-set size (Section III-B) and extend our code construction to provide forward error correction (Section III-C).

A. Single-bit Messages

Our construction leverages properties of erasure correcting codes [9] and proceeds as follows: select a linear code of length N, minimum distance $d_{min} = \min\{2K+1, N+1\}$, and redundancy ℓ, with ℓ as small as possible; consider the $\ell \times N$ parity check matrix $\mathbf{H}$ [9]; assign to each source a different column of $\mathbf{H}$, which corresponds to a one-dimensional subspace of the ℓ-dimensional space. This code results in each active source generating a single vector of length ℓ: to communicate message m_1, the source generates a vector with all zeros; to communicate message m_2, the source generates the vector that corresponds to its column of $\mathbf{H}$.

This code is identifiable because of a well known property of the matrix $\mathbf{H}$: given a linear code with minimum distance d_{min}, *any* $d_{min} - 1$ *columns of the parity check matrix* $\mathbf{H}$ *are linearly independent* [9]. For example, for $d_{min} = 2K + 1$, any $2K$ columns of the parity check matrix are linearly independent; thus, if at one round we have $K_1 \le K$ active sources, each sending a different vector u_i, and at another round we have a different set of $K_2 \le K$ active sources, each sending a vector v_j, then

$$v_1 + v_2 + \cdots + v_{K_1} \ne u_1 + u_2 + \cdots + u_{K_2} \tag{2}$$

This inequality is a direct consequence of the fact that any $2K$ vectors are linearly independent: indeed, if (2) was an equality, there would be $2K$ or fewer linearly dependent vectors. Hence, every possible combination of subspaces generated by the active sources results in a distinct union subspace, which means that our code is identifiable.

Note that we never need to have minimum distance greater than $N+1$, since there exist N nodes and, thus, we can have at most N distinct vectors appearing in (2). This implies that, for all the cases where $N \geq K \geq \frac{N}{2}$, *i.e.*, at least half of the nodes are active, we need $d_{min} = N+1$ and we can select w.l.o.g. the full rank $N \times N$ parity matrix $\mathbf{H}$ to be the identity matrix. In this case, source i generates a vector with 1 at position i and 0 elsewhere. It is easy to see that this code has the optimal length: since each packet has to convey N independent unit entropy source messages (*i.e.*, one bit from each source), it cannot have length less than N.

The scalability of our code depends on how ℓ (the size of the vector generated by each source) depends on N (the size of the network) and K (the maximum number of vectors that can get combined) as N grows, which turns out to be $\mathcal{O}(K \log N)$. This is related to a well studied problem in coding theory, namely, for a given code length N, and a given minimum distance $2K+1$, what are upper and lower bounds on the number of codewords $A(N,K)$ this code can have [9]. Using the Gilbert-Varshamov lower bound and the sphere packing upper bound [9], for $K < \frac{N}{4}$, we get that

$$\ell \leq N H_q(\frac{d_{min}-1}{N}), \text{ and} \tag{3}$$
$$\ell \geq N H_q(\frac{d_{min}-1}{2N}) - \frac{1}{2}\log_q\left(4(d_{min}-1)(1-\frac{d_{min}-1}{2N})\right)$$

where $H_q(\cdot)$ is the q-ary entropy function, namely, $H_q(p) = p\log_q(q-1) - p\log_q p - (1-p)\log_q(1-p)$. Considering fixed values of K and that $\frac{d_{min}-1}{N} = p = \frac{2K}{N}$, the term $\log_q(1-\frac{2K}{N})$ approaches zero, and the upper bound becomes

$$\begin{aligned} \ell \leq N(&\frac{2K}{N}\log_q(q-1) - \frac{2K}{N}\log_q\frac{2K}{N} \\ &- (1-\frac{2K}{N})\log_q(1-\frac{2K}{N})) \qquad (4) \\ \approxeq\ & 2K\log_q N + 2K\log_q\frac{(q-1)}{2K} \qquad (5) \end{aligned}$$

where $2K\log_q\frac{(q-1)}{2K}$ is a constant term. Hence, the behavior of the bound determined by the term $K\log_q N$. A similar argument can be made for the lower bound.

Figure 2 plots the exact values of these bounds from (3) as a function of N, for $K=2$ and $K=20$. We can see that the vector length l resulting from our code is a fraction of the network size N that goes to zero as the ratio $\frac{d_{min}-1}{N} = \frac{2K}{N}$ goes to zero; we conclude that our code is scalable, in the sense that the vector length does not increase proportionally to the network size, but, instead, more slowly, as a function of the maximum number of combined packets. Again, it is easy to see that this code length is optimal: indeed, otherwise, we would be able to construct codes with the required error correcting capabilities and smaller length.

Figure 3 illustrates the benefits of joint identity-message coding over aggregation with respect to busiest-node transmissions. Consider a tree similar to the one in Figure 1, where N sources connect to a sink A through a single link BA. Assume that at most K sources are active, each communicating a single-bit message. Aggregation requires $K\log_2 N$ identity bits to traverse the link BA, whereas coding requires a number of bits ℓ bounded according to (3). Figure 3 shows that coding results in significantly lower load on link BA.

The difference can be interpreted as follows: with coding, to each set of K sources corresponds a specific coded packet that is received by the sink. With aggregation, the K source packets may be aggregated in an arbitrary order, and all $K!$ possible permutations convey the same message to the sink; because this ordering conveys no information, we lose $\log_2(K!)$ bits, which equals the gap between the coding lower bound and aggregation in the plot. Note that the currently best found codes closely follow the lower bound.

Identifiable Codes:

- Let $\mathbf{H}$ be the $\ell \times N\Delta$ parity check matrix of a binary code with minimum distance $\min\{2K\Delta+1, N\Delta+1\}$
- Assign to source i the subspace Π_i spanned by the $(i-1)\Delta+1$ to $i\Delta$ columns of $\mathbf{H}$
- Select $\mathcal{C}_i = \{\pi_i \mid \pi_i \subset \Pi_i, \dim(\pi_i) = 1\}$
- For single-bit messages $\Delta = 1$.

Identifiable Codes for Error Protection:

- Same steps 1 and 2 as in identifiable codes
- Select $C_i = \{\pi_i \mid \pi_i \subset \Pi_i, \dim(\pi_i) = d\}$ with $D(C_i) > 2r_i$

B. General Case

We now consider the general case, where each source communicates one of M messages. The only difference from the single-bit-message case is that, instead of allocating a single column of the matrix $\mathbf{H}$ to source i, we allocate to it Δ columns that span a subspace Π_i; source i can use any sub-subspace within Π_i as a codeword. For simplicity, we only consider the case where source i uses each of the $q^\Delta - 1$ one-dimensional subspaces within Π_i to communicate one of $M = q^\Delta - 1$ different messages. In this particular case, the code results in each active source generating a single vector of length ℓ. In principle, however, we can also use multi-dimensional subspaces as codewords—for instance, any code design method from [10], using Π_i as our original space.

Each active source now generates one vector of length ℓ, which is a linear combination of at most Δ columns of the matrix $\mathbf{H}$. Since at most K vectors can get combined, the sink will receive vectors that are linear combinations of at most $K\Delta$ columns of $\mathbf{H}$. Thus, provided the minimum distance of the code is greater than $2K\Delta+1$, two received vectors will be equal if and only if the set of active users and their messages

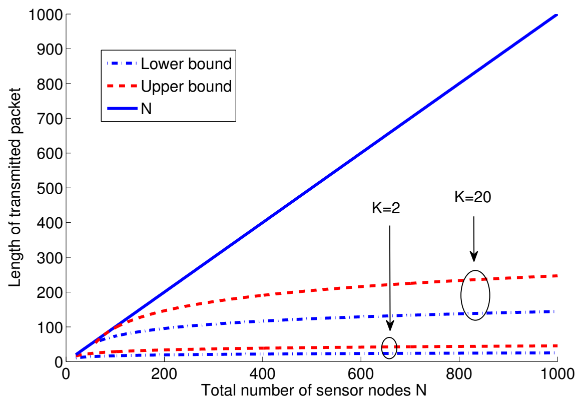


Fig. 2. Bounds on the length ℓ of generated binary vectors ($q = 2$) when $K = 2$ and $K = 20$ sources get combined, as a function of the number of sensor nodes N.

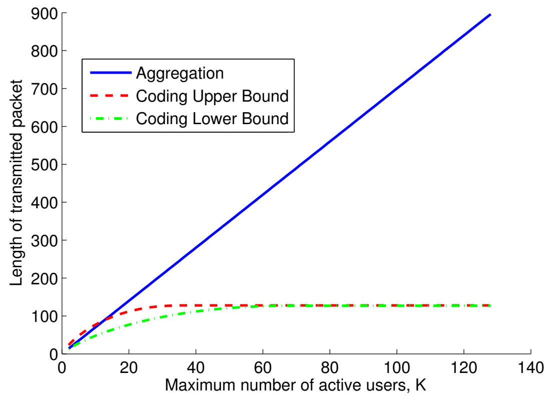


Fig. 3. Comparison of the maximum transmission load for aggregation and coding for $N = 128$, as a function of K.

are the same, so our code is identifiable. As before, given the total number of sources equals N, we can never have a combination of more than $N\Delta$ columns of $\mathbf{H}$, leading to the second upper limit in the required minimum distance.

C. Error Resilience

Our code construction can be naturally extended to provide forward error correction. Such an approach is well matched to the cases where feedback cannot be readily used or sensors fail (and could not retransmit anyway). To achieve this, we need to introduce redundancy into the transmitted information, by separating the subspaces chosen as codewords by a certain "distance." We define the minimum distance of the codebook $\mathcal{C}_i$ as the closest two subspaces from this codebook can get. More formally,

$$D(\mathcal{C}_i) \triangleq \min_{\pi_\alpha,\pi_\beta\in\mathcal{C}_i:\pi_\alpha\neq\pi_\beta} d(\pi_\alpha,\pi_\beta), \qquad (6)$$

where $d(\pi_\alpha,\pi_\beta)$ was defined in (1). Define the union subspace code $\mathcal{C}$ as

$$\mathcal{C} \triangleq \left\{\pi^{(1)}+\cdots+\pi^{(N)} \middle|\ \pi^{(i)}\in\mathcal{C}_i,\ i=1,\ldots,N\right\}. \qquad (7)$$

The following theorem relates the minimum distance of code $\mathcal{C}$ to the error and erasure correction capability of codebooks $\{\mathcal{C}_i\}$ under minimum distance decoding.

Theorem 1: Assume an identifiable code $\{\mathcal{C}_i\}$ is used for transmission over the network. Let $\pi^{(i)}\in\mathcal{C}_i$, $i=1,\ldots,N$, be transmitted and π_R be received. Assume at most r_i vectors get erased from source i and at most t corrupted vectors are injected in the network. Then, if

$$2\left(t+\sum_{i=1}^N r_i\right) < D(\mathcal{C}), \qquad (8)$$

a minimum distance decoder allows the receiver to recover the sent subspaces for each source.

Proof: Define $\pi_S=\pi^{(1)}+\cdots+\pi^{(N)}$ and $\hat{\pi}^{(i)}=\mathcal{E}_{r_i}(\pi^{(i)})$ for $i=1,\ldots,N$, where $\mathcal{E}_r(\pi)$ is an erasure operator that erases randomly from π at most r dimensions. Then for $\pi_S = \pi^{(1)}+\cdots+\pi^{(N)}, \hat{\Pi}=\hat{\pi}^{(1)}+\cdots+\hat{\pi}^{(N)}$,

$$\begin{aligned} d(\pi_R,\pi_S) &\overset{\text{(a)}}{\le} d(\pi_S,\hat{\Pi})+d(\pi_R,\hat{\Pi}) \\ &\overset{\text{(b)}}{\le} \sum_{i=1}^N d(\pi^{(i)},\hat{\pi}^{(i)})+d(\pi_R,\hat{\Pi}) \le \sum_{i=1}^N r_i + t, \end{aligned}$$

where (a) follows from the triangle inequality and (b) follows from Lemma 1. For another codeword $\pi'_S\neq\pi_S$ in $\mathcal{C}$ we have $D(\mathcal{C})\le d(\pi_S,\pi'_S)\le d(\pi_S,\pi_R)+d(\pi_R,\pi'_S)$.
Combining these two inequalities we can write

$$d(\pi_R,\pi'_S)\ge D(\mathcal{C})-d(\pi_S,\pi_R)\ge D(\mathcal{C})-\left(t+\sum_{i=1}^N r_i\right).$$

So if the inequality (8) holds, then $d(\pi_R,\pi'_S) > d(\pi_R,\pi_S)$ and a minimum distance decoder would select π_R. Because $\{\mathcal{C}_i\}$ is an identifiable code, the receiver is able to decompose π_R uniquely and find the original messages. ■

Lemma 1: Suppose π_i, $i=1,\ldots,N$, are subspaces of some vector space W. Assume $\hat{\pi}_i\subseteq\pi_i$ for $i=1,\ldots,N$. Then we have

$$d(\pi_1+\cdots+\pi_N,\hat{\pi}_1+\cdots+\hat{\pi}_N)\le\sum_{i=1}^N d(\pi_i,\hat{\pi}_i).$$

IV. Theoretical Performance Comparison

We now compute theoretical performance bounds for the case of error-free communication, *i.e.*, if we assume that each packet transmitted by a node is successfully received by its parent (without any retransmissions). As performance metrics, we consider the average and maximum codeword used by the communication protocols we compare. We assume that we are provided with a no-cost character that can signal the end of a string, hence, we can use non-prefix-free variable-length packets without paying any additional performance cost.

We compare three communication protocols: (i) *Aggregation*, as defined in Section II-C; (ii) *Coding*, as defined in Section III; and (iii) *Optimal*, which is defined as follows. Each node that acts as a relay maps each possible tuple aggregate A to a binary string, such that the average length of the

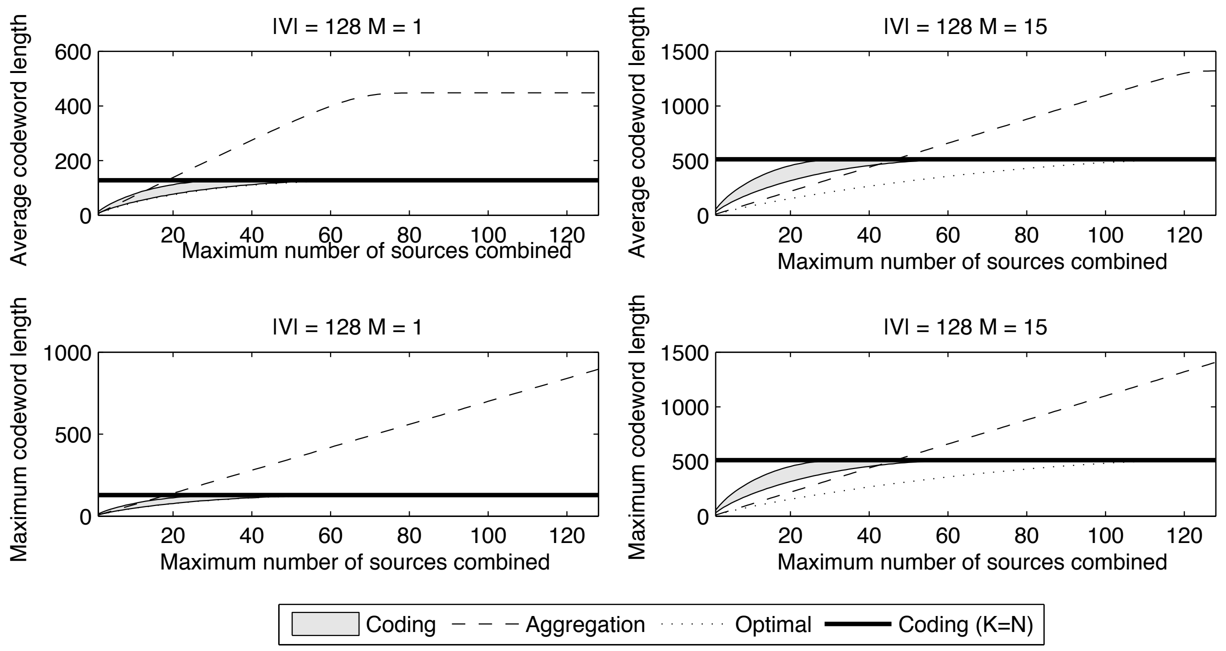


Fig. 4. Average and maximal number of bits for aggregation, coding and the optimal scheme, for a sensor network with $N = 128$ source nodes, as a function of the maximum number K of sources combined in each aggregate.

used codewords is minimal. This allows arbitrary complexity operations at network nodes, as well as variable-length coding.

We here provide the calculations for the average cost (codeword length) for the above three cases; the calculations for the maximum cost are very similar. In all three cases, the goal is to design a codebook that allows each relay to transmit its observed tuple aggregate to the next hop towards the sink. We assume that we have no a-priori topological information, *i.e.*, the coding scheme should support transmission of all possible tuple aggregates from each relay. We thus compute the average value for a uniform distribution over all A, since any tuple aggregate A can be observed. Recall from Section II-C that a tuple aggregate A is the set of tuples that are observed by a relay and correspond to a set of sources $I(A)$.

Given a *fixed set* of sources $I(A)$, the number of possible tuple aggregates is $M^{|I(A)|}$. Thus, given a *fixed number of sources* $|I(A)|$, we can build a number of tuple aggregates equal to $\binom{N}{|I(A)|}M^{|I(A)|}$. The total number of possible tuple aggregates is therefore:

$$\sum_{i=0}^{N}\binom{N}{i}M^i = (M+1)^N.$$

If at most $K \leq N$ sources are contained in each tuple aggregate, each relay must support the transmission of $T^{(K)} = \sum_{i=0}^{K}\binom{N}{i}M^i$ different tuple aggregates.

The performance of aggregation is in this case:

$$\mathrm{AVG}_{\mathrm{bits}}^{\mathrm{agg}}(N,M,K) = \frac{\sum_{i=0}^{K}\binom{N}{i}M^i \cdot i\left(\lceil\log_2(N)\rceil + \lceil\log_2(M)\rceil\right)}{T^{(K)}}$$

The performance of the optimal scheme is:

$$\mathrm{AVG}_{\mathrm{bits}}^{\mathrm{optimal}}(N,M,K) = \frac{\sum_{i=0}^{l-1} i\cdot 2^i + (T^{(K)} - (2^{l(K)}-1))\cdot l}{T^{(K)}}$$

where in this case $l(K)$ is defined as:

$$2^{l(K)}-1 \leqslant T^{(K)} \leqslant 2^{l(K)+1}-1 \Rightarrow l(K) = \left\lceil \log_2\left(T^{(K)}+1\right)\right\rceil - 1.$$

For coding, we can apply the same formula as above using a different $l(K)$ value. By using the bounds described in Section III-B when using field size 2 and $K < N/2$, we have in this case that

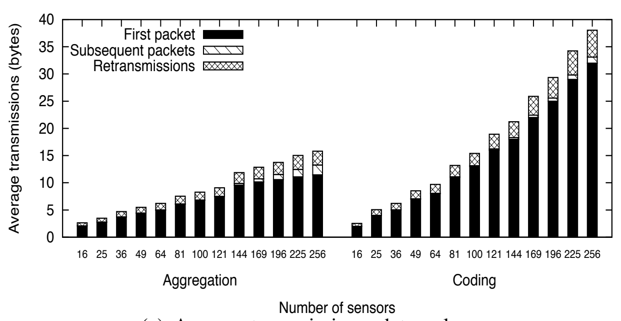
$$\log_2\left(\sum_{i=0}^{2K\Delta} N_i\right) \geqslant l(k) \geqslant \log_2\left(\sum_{i=0}^{K\Delta} N_i\right),$$

where $N_i \triangleq \binom{N\lceil\log_2(M+1)\rceil}{i}$. Note that when $K \geq N/2$ the length of the short codes is $N\lceil\log_2(M+1)\rceil$.

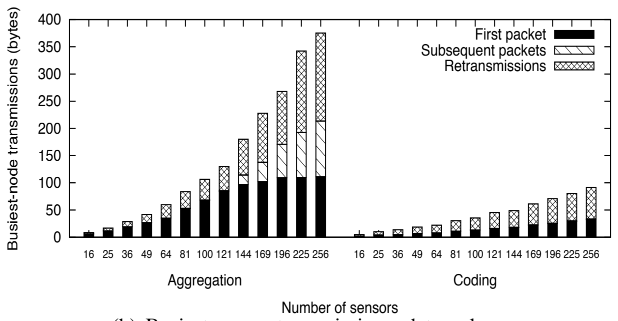
Figure 4 shows the performance of the aggregation, coding, and optimal protocols, in terms of maximum and average codeword length, for a fixed number of nodes $N = 128$, as a function of the maximum number K of sources combined in a tuple aggregate. We observe that coding performs better than aggregation and close to the optimal protocol for small values of M, while its performance deteriorates for larger values of M and small values of K. The plots also depict with a bold line the performance of coding when we allow all possible source combinations (*i.e.*, when $K = N$). This performance deterioration is due to the fact that codes are constrained to employ fixed-length packets and simple operations at network nodes. If higher complexity is allowed, we can opt for optimal codes that significantly outperform aggregation at all regimes. We note that, as K increases, the performance of aggregation is significantly worse than the performance of coding.

V. Simulations

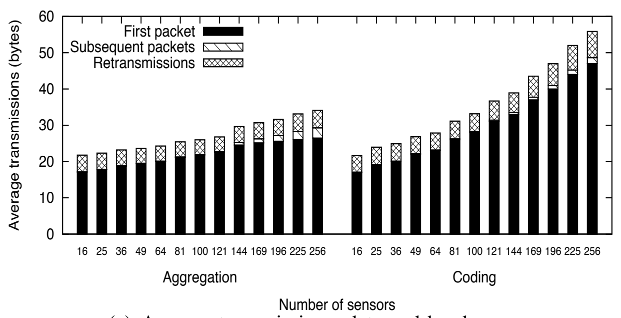
We implemented the aggregation protocol from Section II-B and the joint identity-message coding protocol (we will



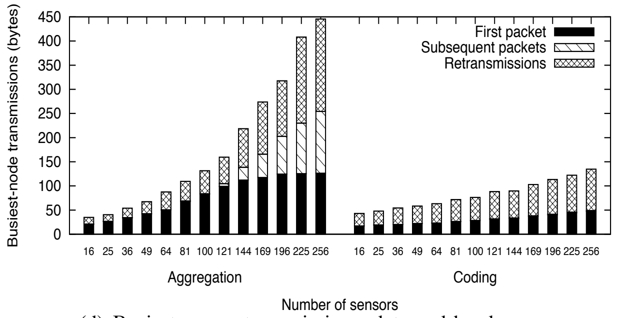
(a) Average transmissions, data only



(b) Busiest-sensor transmissions, data only



(c) Average transmissions, data and headers



(d) Busiest-sensor transmissions, data and headers

Fig. 5. Aggregation and coding performance as a function of network size, when all nodes in the network are active sources. The top graphs are computed by taking into account only the data bytes, *i.e.*, the bytes that carry the identities and messages of the active sources. The bottom graphs take into account both the bytes that carry data and those that carry protocol headers. Both aggregation and coding incur 15 bytes of header overhead per packet. This includes MAC and CTP headers.

call it the "coding protocol" for brevity) from Section III as TinyOs [11] modules and compared their performance using the TOSSIM simulator [2]. We now describe our testing setup and results.

A. Setup

We simulate a fixed network of MICAz sensors [12] located in a parking structure, where the sensors are placed on a square grid with the sink located on the upper left corner. We place each node at 8 meters from its closest neighbors (this choice is justified below). We use the channel parameters for a parking structure reported in [13].

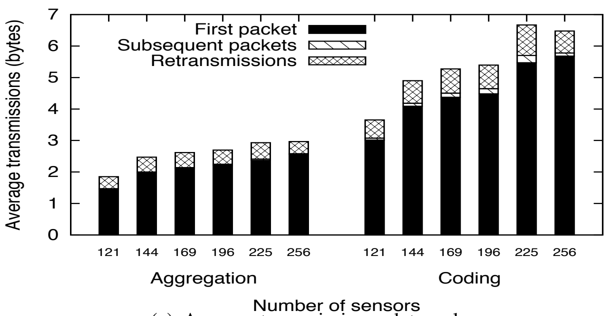
Our simulated network operates in rounds. In each round, some or all of the nodes in the network act as active sources, *i.e.*, communicate their identity and a 1-bit message to the sink. We do not experiment with more than 256 nodes, because we found that connecting more nodes to a single sink causes congestion and packet loss around the sink (no matter what protocol we use to send data to the sink).

We use the default CSMA based MAC layer included in TinyOs. We rely on standard MAC-layer acknowledgments and retransmissions for error protection, *i.e.*, we do not use our coding-based error correction from Section III-C. We allow a maximum payload of 120 bytes and a maximum number of 30 retransmissions per packet (the default value of the MAC-layer implementation that comes with TinyOs), although, in practice, we rarely observe more than 5 retransmissions. The channel conditions and network congestion in our simulations are such that each packet has to be retransmitted on average 1.5 times to successfully reach the transmitter's parent. We chose this number, because it is consistent with reports from deployed sensor networks [8]. We should note that this number makes sense, given that sensor networks are designed to be energy-efficient, hence it is reasonable for nodes to be placed close enough to avoid multiple retransmissions in the common case.

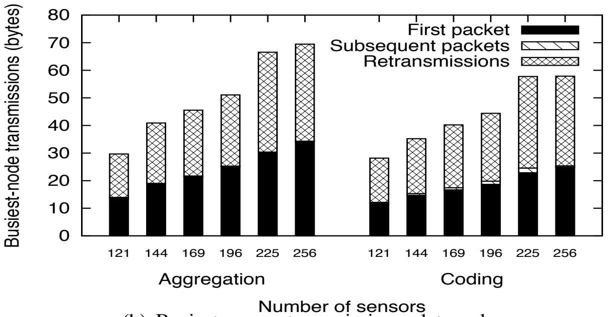
We rely on the Collection Tree Protocol [8] for creating and maintaining a spanning tree rooted at the sink. In both aggregation and coding, each node tries to send only one packet in each round, *i.e.*, receive data from all its children, combine them into one packet, and send that to its parent. This is achieved in the following way: In each round, each node records the number of its children, *i.e.*, the number of nodes that sent data to it. In the next round, the node sends a packet to its parent as soon as it has received one packet from each recorded child, or a timer has expired. It is possible that the node receives a packet *after* it has already sent one packet to its parent; such "delayed packets" are immediately forwarded to the receiver's parent.

We place each node at 8 meters from its closest neighbors, because this setting allows more than 99% of the transmitted packets to reach the sink. *I.e.*, even though we use MAC-layer retransmissions and CTP to keep our network connected in the face of channel changes, if we place our sensors more than 8 meters from one another, at least 1% of the transmitted packets fail to reach the sink, no matter what protocol we use (even if we use plain CTP without aggregation or coding).

We compare aggregation and coding in terms of two metrics: (i) "Average transmissions" is the total number of bytes transmitted by all nodes during the experiment, divided by the number of nodes and the number of rounds. This metric



(a) Average transmissions, data only



(b) Busiest-sensor transmissions, data only

Fig. 6. Aggregation and coding performance as a function of network size, when 20% of the nodes in the network are active sources. The graphs are computed by taking into account only data bytes, *i.e.*, bytes that carry the identities and messages of the active sources.

shows how the two protocols compare in terms of total energy spent by all nodes, where we expect coding to do worse than aggregation. (ii) "Busiest-node transmissions" is the total number of bytes transmitted by the node that transmits the most bytes during the experiment, divided by the number of rounds. This metric shows how the two protocols compare in terms of energy spent by the busiest node, where we expect coding to do (significantly) better than aggregation.

We break down each result in three non-overlapping elements: (i) "First packets" consist of the first packet sent by each node to its parent during each round. (ii) "Subsequent packets" consist of the packets sent by each node after the node has already sent one packet to its parent in the current round. These may include the delayed packets mentioned above and/or packets due to fragmentation, *i.e.*, created when a node has more data to send to its parent than can fit in a single packet. (iii) "Retransmissions" refer to MAC-layer retransmissions.

We consider subsequent packets separately, because they can have a significant performance impact on coding, whereas they do not affect aggregation. To understand why, consider the network of Figure 1 and suppose it runs the coding protocol, *i.e.*, each of the 8 leaf nodes sends an 8-bit vector to node B, which XORs all vectors and sends the resulting 8-bit vector to the sink. Now suppose that node S_8 sends out its vector *after* B has already sent one vector to the sink; as a result, B has to forward S_8's vector to the sink as is, which means that B now sends a total of 2×8 bits of data to the sink (twice as it would send if there were no delayed vectors). In contrast, if the same network runs the aggregation protocol, each of the 8 leaf nodes sends a 4-bit $\{$identity, message$\}$ tuple to B, which forwards each tuple to the sink. So, B sends 8×4 bits of data to the sink, independently from whether it manages to combine all tuples into a single packet or not.

So, we expect subsequent packets to reduce, to some extent, the advantage of coding over aggregation in terms of busiest-node transmissions. The question is by how much, and this is what we study with our experiments.

B. Results

Figure 5 shows the average and busiest-node transmissions as a function of network size, when all nodes in the network are active sources. As expected, coding results in more average transmissions and significantly fewer busiest-node transmissions than aggregation. For instance, in a 100-node network, aggregation causes the average node to transmit about 10 bytes of data per round and the busiest node to transmit about 100 data bytes per round; coding causes the average node to transmit about 15 bytes of data per round and the busiest node to transmit about 40 bytes of data per round.

The effect of subsequent packets on coding performance is insignificant. Of course, this is because, in our implementation, each node waits to receive from as many children as possible before sending to its parent. However, the point of doing an actual TinyOs implementation and TOSSIM experiments was precisely to show that, in a fixed, realistic network, it *is* feasible, in most rounds, for the nodes to estimate the number of their children, hence, it is also feasible to practically eliminate delayed packets.

We observe that, when we have 144 or more nodes and we use aggregation, the busiest node incurs a relatively large number of subsequent packets and retransmissions. This is explained as follows. When we use aggregation, the busiest node is a neighbor of the sink that acts as a relay to a large number of other nodes. When we have more than 144 nodes, the busiest node receives, in each round, more data than can fit into a single packet, hence must send subsequent packets to its parent (the sink). The large number of retransmissions is due to the fact that the busiest node has to send relatively large packets, which are susceptible to transmission errors.

Figure 6 shows the average and busiest-node transmissions as a function of network size, when only 20% of the nodes in the network are active sources. This is a "bad" scenario for coding, as there are relatively few active sources, hence, even the busiest node incurs a relatively low load, which means that there is not much room for improvement through coding. *E.g.*, in a 256-node network, where only 64 of the nodes are active sources, aggregation causes the average node to transmit about 3 bytes of data per round and the busiest node to transmit about 70 bytes of data per round; coding causes the average node to transmit about 7 bytes of data per round and the busiest node to transmit about 60 bytes of data per round.

VI. Related Work

Recently, techniques inspired from coding and network coding have been successfully used to harness the broadcasting capabilities of the wireless medium [14], [15], [16], [17], [18], implement intelligent in-network storage [19], and provide resilience in lossy environments [20]. These coding techniques are not suitable for identity-aware sensor networks, as they are developed with a focus on data dissemination (as opposed to joint identity/message delivery).

Our proposed coding scheme relies on subspace codes, which have been studied in the context of non-coherent communication over fading point-to-point wireless channels [21], quantum communication [22], and, more recently, network coding, to provide error and erasure correction [10], [23]. The previous constructions address the single-source case, do not encode the source identity, and are designed for large-packet transfers. We develop constructions that incorporate in the code the identity of multiple sources with low communication overhead and are targeted to (very) small-packet transfers. Additionally, our constructions obey the constraint that each sensor node transmits the same number of packets, independently of its location in the network, which is not the case for the constructions in [10], [24], [23].

Significant research effort has been invested in reducing sensor-communication overhead through distributed, in-network data aggregation. The proposed techniques exploit data compression [5], [4], [25] or calculate functions of the observed measurements such as their average [25]. None of these is applicable in the case where node identities form the bulk of the data. For small packet lengths, the optimal (non-coding) approach is aggregation, *i.e.*, to package, at each node, all the received identities and messages into a single packet. As discussed in more detail in the paper, this requires in-network content processing and, most importantly, results in unequal-length packets whose size increases significantly as we approach the sink; hence, energy consumption is unequally distributed among sensor nodes, in particular, concentrated on centrally located "bottleneck nodes" that can, as a result, fail and disrupt network operation.

Finally, our proposed approach does not fall in the framework of synopsis operation [26] for several reasons, most importantly because (i) it does not guarantee the duplication insensitivity property and (ii) our sources may send multiple instead of one packets for each periodic measurement.

VII. Conclusions

We have formulated the paradigm of identity-aware sensor networks to capture applications where the identities of the sensors form the bulk of the communicated data. We have proposed a communication protocol for such networks, where sensor identities and measurements are jointly encoded in fixed-size vectors; to the best of our knowledge, this is the first such approach. Its benefits consist of balancing the transmission load across all nodes in the network (important for networks that are periodically recharged through natural resources), low-complexity network operations, and graceful incorporation of error resilience.

References

[1] Jiang, J. Polastre, and D. Culler. Perpetual Environmentally Powered Sensor Networks. In *International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[2] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS applications. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[3] G. Pottie and W. Kaiser. *Principles of Embedded Networked Systems*. Cambridge University Press, 2005.

[4] A. Scaglione and S. D. Servetto. On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks. *Wireless Networks*, 11(1-2):149–160, 2005.

[5] S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed Compression in a Dense Sensor Network. *IEEE Signal Processing Magazine*, 19(2):51–60, March 2002.

[6] D. Tulone and S. Madden. PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks. In *European Workshop on Wireless Sensor Networks*, 2006.

[7] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.

[8] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection Tree Protocol. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[9] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library, 1977.

[10] R. Koetter and F. Kschischang. Coding for Errors and Erasures in Network Coding. In *IEEE International Symposium on Information Theory (ISIT)*, 2007.

[11] Tinyos. http://www.tinyos.net/.

[12] MICAz sensors.

[13] K. Sohrabi, B. Manriquez, and G. Pottie. Near Ground Wideband Channel Measurements at 800-1000MHz. *IEEE Transactions on Vehicular Technology*, 1:571–574, May 1999.

[14] Y. Wu, P. A. Chou, and S. Y. Kung. Minimum-Energy Multicast in Mobile Ad-hoc Networks Using Network Coding. *IEEE Transaction on Communications*, 53(11):1906–1918, November 2005.

[15] S. Katti, S. Gollakota, and D. Katabi. Embracing Wireless Interference: Analog Network Coding. In *ACM SIGCOMM*, 2007.

[16] C. Fragouli, J. Widmer, and J.-Y. L. Boudec. A Network Coding Approach to Energy Efficient Broadcasting: From Theory to Practice. In *IEEE INFOCOM*, 2006.

[17] S. Zhang, S. Liew, and P. Lam. Physical Layer Network Coding. In *ACM MOBICOM*, 2006.

[18] S. Sengupta, S. Rayanchu, and S. Banerjee. An Analysis of Wireless Network Coding for Unicast Sessions: The Case for Coding-Aware Routing. In *IEEE INFOCOM*, 2007.

[19] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth Codes: Maximizing Sensor Network Data Persistence. In *ACM SIGCOMM*, 2006.

[20] M. Ghaderi, D. Towsley, and J. Kurose. Network Coding Performance for Reliable Multicast. In *Military Communication Conference (MILCOM)*, 2007.

[21] F. Oggier, N. J. A. Sloane, S. N. Diggavi, and A. R. Calderbank. Non-Intersecting Subspaces with Finite Alphabet. *IEEE Transactions on Information Theory*, 51(12):4320–4325, December 2005.

[22] A. R. Calderbank, E. M. Rains, P. M. Shor, and N. J. A. Sloane. Quantum Error Correction via Codes over GF(4). *IEEE Transactions on Information Theory*, 44(4):1369–1387, July 1998.

[23] C. Fragouli M. Jafari Siavoshani, S. Mohajer and S. Diggavi. Capacity of non-coherent network coding. *ISIT*, 2009.

[24] D. Silva and F. R. Kschischang. Using rank-metric codes for error correction in random network coding. *IEEE International Symposium on Information Theory (ISIT)*, 2007.

[25] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[26] S. Nath, P. B. Gibbons, S. Seshan, Z. R. Anderson, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. *Proc. SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, page 250, 2004.

Lorenzo Keller is pursuing a Ph.D. in the School of Computer and Communication Sciences, EPFL, Switzerland. He received his bachelor and master degrees in Communication Systems from the same institution in 2005 and 2007 respectively. His research is on theoretical and practical aspects of wireless networks and, in particular, on improving the performance, reliability and energy efficiency of sensor networks.

Mahdi Jafari Siavoshani received the Bachelor degree in Communication Systems with a minor in Applied Physics at Sharif University of Technology, Tehran, Iran, in 2005. He was awarded an Excellency scholarship from EPFL, Switzerland, to study a master degree in Communication System finished in 2007. He is currently a PhD student at the same university. His research interests include network coding, coding and information theory, wireless communications, and signal processing.

Christina Fragouli is an Assistant Professor in the School of Computer and Communication Sciences, EPFL, Switzerland. She received the B.S. degree in Electrical Engineering from the National Technical University of Athens, Athens, Greece, in 1996, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of California, Los Angeles, in 1998 and 2000, respectively. She has worked at the Information Sciences Center, AT&T Labs, Florham Park New Jersey, and the National University of Athens. She also visited Bell Laboratories, Murray Hill, NJ, and DIMACS, Rutgers University. From 2006 to 2007, she was an FNS Assistant Professor in the School of Computer and Communication Sciences, EPFL, Switzerland. She served as an editor for IEEE Communications Letters, and is currently serving as an editor for IEEE Transactions on Information Theory, IEEE Transactions on Communications and for Elsevier Computer Communications. She received the Fulbright Fellowship for her graduate studies, the Outstanding Ph.D. Student Award 2000-2001, UCLA, Electrical Engineering Department, the Zonta award 2008 in Switzerland, and the Young Investigator ERC grant in 2009. Her research interests are in network information flow theory and algorithms, network coding, and connections between communications and computer science.

Katerina Argyraki is a researcher with the Operating Systems group in the School of Computer and Communication Sciences, EPFL, Switzerland. She works on network architectures and protocols with a focus on denial-of-service defenses and accountability. She received her undergraduate degree in Electrical and Computer Engineering from the Aristotle University, Thessaloniki, Greece, in 1999, and her Ph.D. in Electrical Engineering from Stanford University, in 2007.

Suhas Diggavi received the B. Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, India, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1998.

After completing his Ph.D., he was a Principal Member Technical Staff in the Information Sciences Center, AT&T Shannon Laboratories, Florham Park, NJ. Since then he has been in the faculty of the School of Computer and Communication Sciences, EPFL, where he directed the Laboratory for Information and Communication Systems (LICOS). He is currently a Professor, in the Department of Electrical Engineering, at the University of California, Los Angeles. His research interests include wireless communications networks, information theory, network data compression and network algorithms.

He is a recipient of the 2006 IEEE Donald Fink prize paper award, 2005 IEEE Vehicular Technology Conference best paper award and the Okawa foundation research award. He is currently an editor for ACM/IEEE Transactions on Networking and IEEE Transactions on Information Theory. He has 8 issued patents.