Identity Aware Sensor Networks

ARNI technical report 02, June 2008

Ecole Polytechnique Fédérale de Lausanne

Lorenzo Keller, Mahdi Jafari Siavoshani,
Katerina Argyraki, Christina Fragouli, Suhas Diggavi

ABSTRACT

In a significant class of sensor-network applications, the identities of the reporting sensors are an essential part of the reported information. For instance, in environmental monitoring, the goal is to reconstruct physical quantities over space and time; these quantities are sampled by the sensors, and the source identity associated with each measurement is necessary for the spatial and temporal reconstruction. In many practical scenarios, source identities constitute the bulk of the communicated data, whereas the message itself can be as small as a single bit. In these scenarios, the traditional network-protocol paradigm of separately specifying the source identity and the message in distinct fields leads to inefficient communication.

In this paper, we re-examine this traditional data separation and propose a scheme for joint identity-message encoding; we use this scheme to design a new efficient collection protocol for identity-aware sensor networks. Compared to conventional data collection, our protocol reduces the amount of traffic in the network at least by a factor of two (up to an order of magnitude, in lossy environments), while its performance scales better with network complexity; we show these results both through theoretical analysis and extensive simulations.

1. 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: First of all, any intelligent source-identity encoding could come at the cost of increased network complexity, as intermediate network nodes often need to determine the packet's source as part of the forwarding process. Most importantly, 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 (ad-hoc) 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—temperature exceeds a safety threshold, a perimeter is violated, soil or water is contaminated etc.; in other cases, they are used to track (typically small) incremental changes at different locations, *e.g.*, the evolution of snow height at different 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 describe such paradigms as *identity aware* sensor networks.

Identities are not of the same nature as messages: for a fixed node, the identity is a constant number that does not change with every transmission, in contrast to the messages that do; we leverage this to develop a new method for the efficient representation and communication of identities, with the purpose of conserving the limited sensor-network energy resources.

To the best of our knowledge, our work is the first to develop an energy-efficient collection protocol tailored to identity aware sensor networks. Our approach has two key elements: (i) Instead of specifying its identity and measurement in separate fields, each reporting node *jointly encodes* the two using a set of fixed-size vectors, which are encapsulated into separate packets. (ii) Intermediate nodes perform *in-network data combination, i.e.*, when they receive vectors from multiple sources, they combine them and relay only the resulting combined vector (which has the same fixed size) to the sink. The combination is done without requiring the intermediate nodes to understand the contents of the packets.

The insight for using joint encoding is straightforward: when source identities form the bulk of the communicated data, it makes sense to consider them as part of any encoding/compression scheme. The need for in-network data combination is related to link-layer overhead: sensor net-

works typically use variations of the IEEE 802.15.4 frame format, which dictates 17 bytes of physical- and MAC-layer header and footer; when the measurement reported by each sensor consists of a few bits, using one frame per measurement means paying a 17-byte cost to transmit only a few bits of useful information. Hence, we design our joint-encoding schemes such that they enable simple, practical in-network combination of multiple measurements into a single frame, without requiring any content inspection.

Our approach is based on subspace encoding (§3): nodes do not convey their identity explicitly, but instead through the choice of their codebook, *i.e.*, the set of vectors they transmit. We exploit the invariance properties of subspaces, such that neither the reporting nodes nor the collecting sink need any knowledge of either network topology or intermediate-node operations. Using this approach, we generate practical codes for different scenarios—good network connectivity, significant loss rate, and networks with a large number of nodes where only few of them are active during each reporting interval.

We incorporate our joint-encoding scheme into an appropriately designed data collection protocol (§4). Our design objectives are:

• *Decentralized Operation:* Each sensor has only local knowledge of network connectivity, *i.e.*, information about its neighbors.

• *Energy Efficiency and Balancing:* We optimize our protocol for reducing the number of sensor transmissions; moreover, we require each sensor to do approximately the same number of transmissions.

• *Error Correction, Scalability and Adaptability*: We want our scheme to gracefully incorporate error protection, scale with the number of sensors, and be flexible so as accommodate specific application requirements.

We implement our data collection protocol as a TinyOs [27] application for TinyNode [30] motes, and evaluate it using the TOSSIM [16] simulator (§5). Compared to conventional data collection, our protocol reduces the number of frame transmissions (and transmitted bytes) at least by two factors (up to an order of magnitude, in lossy environments), while reducing end-to-end message loss. Moreover, it evenly distributes energy consumption among the nodes, reducing the variance of per-node transmissions by several factors.

2. SENSOR IDENTIFICATION

2.1 Applications

We consider sensor networks where each node needs to communicate (i) its identity and (ii) a small (relative to the identity) measurement 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 measurement with a specific sensor. Given our departure from the commonly used paradigm, to motivate our work, we discuss a few examples of applications where our conditions hold, *i.e.*, sensor identity is a critical part, and forms the bulk of the collected data. These are all cases, where the sensors are used to periodically reconstruct the *spatial field* of the physical quantity measured by the sensors, *i.e.*, the variation of that quantity as a function of space [20].

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, under normal operation. 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 could need only a few bits [20].

Spatial Correlation. In other cases, the spatial field under measurement at a *given time*, varies smoothly over space. For instance, the variation of temperature, or pressure etc., is governed by physics which gives a smoothness to the spatial field [20, 22]. We can leverage such smooth variation by having a set of densely deployed sensors communicate only a few bits of information, and using techniques like distributed source coding [21] to still be able to represent the 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 not interested in reconstructing an entire spatial field, only a few "interesting" regions. For example, we might want to examine areas, where the measurements suggest that there is a potential for an avalanche, more closely. In such scenarios, it makes sense to collect data in stages: have each sensor communicate its identity along with few bits of information (just enough to get a coarse representation of the field) and, if something interesting is revealed, query the relevant sensors for more information [5, 24]. In many practical cases, it is enough to have each sensor send a single bit of information (signaling whether a threshold was reached, a perimeter was violated, or an animal was sighted), then query the interesting subset for more precise measurements. Here as well, the sensor identity is a critical part of the information to be conveyed.

2.2 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.

Next, we illustrate this inefficiency with a simple example and introduce the idea of joint identity-message coding.

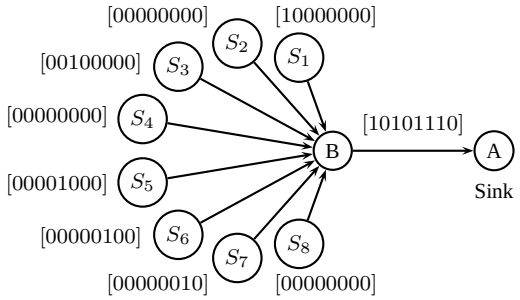


Figure 1: The sources $S_1, \ldots, S_8$ send their id and one bit of information to the sink A through a relay node B.

Consider the simplified network of Fig. 1, where the nodes communicate over an IEEE 802.15.4-compatible link layer. Suppose each node $S_i, i = 1, \ldots, 8$, needs to communicate 1 bit of information to the sink A; it specifies this single bit in a packet and sends it through the intermediate node B. To relay this information to A, B could naively forward it the 8 packets; this would result in 8 wireless frame transmissions, i.e., 8×17 bytes of MAC-layer headers to transmit only 8 bits of information. To avoid this overhead, B could combine all information in a single packet: package the 8 bits in a vector, with the understanding that position i corresponds to the message sent by node S_i; this is the simplest example of using a "code" to represent the identity of a node along with its message. The problem with such in-network coding is that it requires the intermediate node B to understand and process the contents of incoming packets; it would be more practical to develop a coding scheme that operates on an end-to-end basis, *i.e.,* information is always encoded at its source and decoded at the sink, while each node is oblivious to the codes used by other nodes.

Now consider the following alternative: Each node S_i sends out an 8-bit packet with its message encoded in bit i and all other bits set to 0; node B just XORs all incoming packets and sends the resulting 8-bit packet to A. This approach leads to efficient communication on link BA, while keeping node-B functionality simple; the price we pay is a small decrease in efficiency on the S_iB links, which now have to carry 8- (rather than 1-) bit packets, which is insignificant considering the MAC-header overhead[1].

In general, the idea is that each node S_i 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 classical 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 ([4], chapter 14).

Although simple, the network of Figure 1 captures the behavior of all trees (with an arbitrary number of nodes) that connect 8 sources to the sink through the edge BA. For instance, consider the network of Figure 2 and the following two communication protocols: In the first one, each source sends out a packet with a 3-bit identity and 1-bit message specified in separate fields; intermediate nodes simply forward incoming packets towards the sink. In the second protocol, sources jointly encode their identity and message as described above, *i.e.,* source S_i sends an 8-bit packet with its message specified at bit i and all other bits set to zero; each intermediate node XORs all incoming packets and forwards the one resulting packet towards the sink. Note that, using the second protocol, no matter what the tree looks like, for a given set of messages, we will have exactly the same coded packet traversing link BA. Comparing the two protocols,

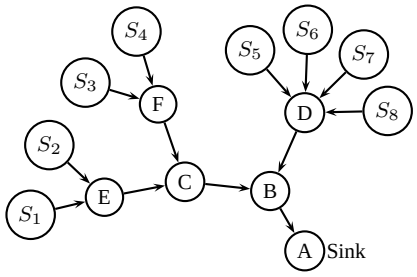


Figure 2: A tree with 8 sources.

the one that uses joint identity-message coding results in a smaller number of packets (13 instead of 28) and a smaller number of information (identity+message) bits (104 instead of 112). Moreover, each node forwards the *same* number of packets and bits, *i.e.,* communication overhead is evenly distributed across the network. This alleviates the problem of depleting the battery of the nodes located close to the sink.

The simple scheme we have described illustrates the basic benefits of joint identity-message coding, but has certain limitations: First, it does not work well in scenarios with heavy network loss (where the single 8-bit packet forwarded from B to A has a high probability of being dropped or in error). Second, it does not scale well to very large networks (in a 500-node network, each node would have to send 500 bits to communicate its single-bit message). We address these issues in Section 3, where we formally present our codes.

2.3 Model and General Approach

We consider a sensor network, with n nodes each of which wants to communicate (i) its identity and (ii) a small measurement, to a collecting sink. The measurement is from a

[1]For IEEE 802.15.4-compatible link layer, each MAC packet has a header of 17 bytes, and we would like to avoid changing in link layer protocols.

set $\mathcal{M}_i$, where[2] $|\mathcal{M}_i| \ll n$.

Each source node i maps its message into a set of vectors and forwards them towards the sink in separate packets. Each intermediate node that receives packets from its neighbors performs linear operations (e.g., XOR) on their contents and forwards the result towards the sink. As a result, the sink, after combining all the received packets, gets a set of vectors $\{\mathbf{y}_i\}$, representable as

$$\mathbf{Y} = \begin{bmatrix} \vdots \\ \mathbf{y}_i \\ \vdots \end{bmatrix} = \sum_{i=1}^{n} \mathbf{G}_i \mathbf{X}_i = \begin{bmatrix} \dots & \mathbf{G}_i & \dots \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{X}_i \\ \vdots \end{bmatrix}, \quad (1)$$

where each $\mathbf{X}_i$ is a matrix having the set of original (row) vectors $\{\mathbf{x}_i^{(j)}\}$ sent by source i as its rows, and $\mathbf{G}_i$ is a "mixing matrix", representing the end-to-end transformation of $\mathbf{X}_i$. That is, the received vectors $\{\mathbf{y}_i\}$ are a linear combination of the original vectors and, hence, lies in their *span*[3].

Neither the sink nor the sources know the mixing matrices $\{\mathbf{G}_i\}$, *i.e.*, the specific set of linear operations performed by the network is unknown to all players. This makes the scheme robust to topology changes. As a result, each source can only communicate information using the *subspace*[4] spanned by its vectors, which is unaffected by the linear operations performed on them. Hence, each source uses a *subspace codebook, i.e.*, maps each message to a set of vectors that span a different subspace. For instance, in Figure 1, source S_1 maps message "1" to vector $[1\ 0\ \dots\ 0]$ (which spans a one-dimensional subspace); as long as no other source uses this subspace, the sink can decode S_1's message, independently from how its original vector is combined with other vectors in the network.

Our goal is to design practical subspace codebooks that enable accurate decoding at the sink (*i.e.*, the sink can easily tell which source sent what). Moreover, it was recently shown, in the context of single-source multicast, that subspace codebooks can provide[5] error correction [15]. We leverage this and combine identity and message information such that the sink can perform decoding even in the presence of adverse network conditions.

2.4 Working with Subspaces

Since in this work the proposed approach extensively relies on use of subspaces, we here give some background material and show that working with subspaces amounts to elementary operations (see standard references like [11] for more details). The impatient reader is encouraged to skip ahead, and consult this Section for notation as needed.

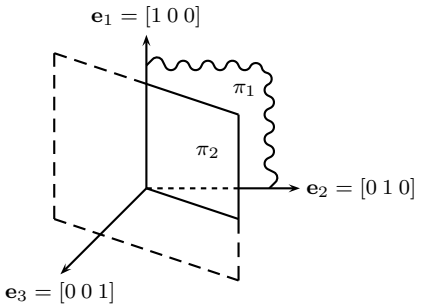


Figure 3: The subspaces $\pi_1 =< \mathbf{e}_1, \mathbf{e}_2 >$ and $\pi_2 =< \mathbf{e}_2 + \mathbf{e}_3, \mathbf{e}_1 >$ in the 3-dimensional space $\mathbb{F}_2^3$.

Information through our network is transferred through the exchange of binary vectors of length ℓ, that belong in the ℓ-dimensional vector space $\mathbb{F}_2^\ell$. We can treat this vector space in exactly the same way as the usual vector space over the reals, the main differences being that we perform XOR instead of real addition operations, and that we have a finite instead of an infinite number of vectors. For example, if $\ell = 3$, the 3-dimensional space $\mathbb{F}_2^3$ contains $2^3 = 8$ vectors.

Given a set of vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$, their *span*, denoted by $< \mathbf{v}_1, \dots, \mathbf{v}_n >$, is the set of all possible linear combinations of these vectors. More precisely,

$$< \mathbf{v}_1, \dots, \mathbf{v}_n >= \{\mathbf{x} : \mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i\}, \quad (2)$$

where the summation is taken over $\mathbb{F}_2$ (XOR) and the scalars $\alpha_i \in \mathbb{F}_2$. This set is a vector space V *generated* by the vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$. Any set of linearly independent vectors that generate a vector space is called a *basis* for this vector space. The dimension of a vector space V, denoted as $\dim(V)$, is the cardinality of a basis, which is the minimum number of vectors needed to be linearly combined in order to create all vectors in the vector space. In our example, the dimension of our space equals $\ell = 3$. A basis consists of the linearly independent vectors

$$\{\mathbf{e}_1 = [1\ 0\ 0], \quad \mathbf{e}_2 = [0\ 1\ 0], \quad \mathbf{e}_3 = [0\ 0\ 1]\}.$$

A subspace is a subset of the vector space V that is a vector space itself. We can think of these subspaces as "planes" that contain the origin. For example, the space $\mathbb{F}_2^3$ (used in Figure 2.4) contains 7 two-dimensional subspaces. One such subspace is $\pi_1 =< \mathbf{e}_1, \mathbf{e}_2 >$. Another is $\pi_2 =< \mathbf{e}_2 + \mathbf{e}_3, \mathbf{e}_1 >$. It also contains 7 one-dimensional subspaces, one corresponding to each non-zero vector. Moreover, the subspace (plane) $\pi_1 =< \mathbf{e}_1, \mathbf{e}_2 >$ contains the three "line" (one-dimensional) sub-subspaces $\pi_3 =< \mathbf{e}_1 >$, $\pi_4 =< \mathbf{e}_2 >$, $\pi_5 =< \mathbf{e}_1 + \mathbf{e}_2 >$. Therefore, we can define subspaces of lower dimension as sub-sub-spaces of higher dimensional subspaces. In the above example, we can see that $\pi_3 \subset \pi_1 \subset \mathbb{F}_2^3 = V$.

We say that two subspaces are *disjoint* if they intersect only trivially at the all zero vector, and that they are *distinct* if they differ in at least one dimension. For example, $\pi_1 =<$

[2]The methods could also be useful when $|\mathcal{M}_i| \approx n$, due to MAC layer overhead, which encourages transmission of smaller number of packets.

[3]The span of a set of vectors are those that are representable as a linear combination of them (see also § 2.4).

[4]A subspace is a subset of the overall space, that forms a vector space itself [11] (see also § 2.4).

[5]Since they considered single source transmission, the issue of conveying identity did not arise.

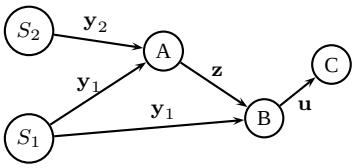


Figure 4: Linear mixing inside a network does not alter the inserted subspace.

$\mathbf{e}_1, \mathbf{e}_2 >$ and $\pi_2 =< \mathbf{e}_2 + \mathbf{e}_3, \mathbf{e}_1 >$ are distinct, while $\pi_1 =< \mathbf{e}_1, \mathbf{e}_2 >$ and $\pi_3 =< \mathbf{e}_3 >$ are disjoint.

As in most of this work we will use the binary field, we will focus our attention on binary operations. However the same ideas go through when we have linear operations over an arbitrary field $\mathbb{F}_q$ and a vector space $\mathbb{F}_q^\ell$. On many occasions we will also give results over $\mathbb{F}_q^\ell$.

Recall that in Section 2.3 we saw that nodes do linear mixing operations, as seen in (1). Now, the critical observation is that such linear operations *preserve the subspaces*. More concretely, consider the situation depicted in Figure 2.4. Let us assume that there are packets represented as vectors $\mathbf{y}_1, \mathbf{y}_2$ inserted by two sources, which could even be over an arbitrary field $\mathbb{F}_q^\ell$. Node A, that receives these two packets would send out a linear combination of them, and therefore, its output, $\mathbf{z}$, would lie in the span of these two vectors, *i.e.*, $\mathbf{z} \in < \mathbf{y}_1, \mathbf{y}_2 >$. Now, suppose that node B receives both $\mathbf{z}$ as well as $\mathbf{y}_1$, its output $\mathbf{u}$, which is a linear combination of $\mathbf{y}_1$ and $\mathbf{z}$, would still lie in $< \mathbf{y}_1, \mathbf{y}_2 >$. This property holds even when operations are over arbitrary fields $\mathbb{F}_q$ (not just binary). Therefore, even though we might get several different vectors in the mixing process, the subspace $< \mathbf{y}_1, \mathbf{y}_2 >$ inserted by the sources remains invariant. This is the property that makes the use of subspaces for encoding robust to the topology of the network and to the linear operations performed at the intermediate nodes.

Given these properties of subspaces, our approach described in § 2.3 shows that we need to encode information through different subspaces. A natural question is whether there are enough subspaces for us to work with. As already seen in the example with $\mathbb{F}_2^3$, there are 7 distinct two-dimensional subspaces. In more generality, it can be shown that $K_\ell(d)$, the number of distinct d dimensional subspaces of $\mathbb{F}_q^\ell$ equals

$$K_\ell(d) = \begin{bmatrix} \ell \\ d \end{bmatrix}_q = \frac{(q^\ell - 1)\cdots(q^{\ell-d+1} - 1)}{(q^d - 1)\cdots(q - 1)}, \quad (3)$$

where $\begin{bmatrix} \ell \\ d \end{bmatrix}_q$ is called the Gaussian number.

We will use two subspace operations. Consider two subspaces π_α and π_β of a vector space V.

• Their *sum*: $\pi_\alpha + \pi_\beta = \{x + y | x \in \pi_\alpha, y \in \pi_\beta\}$ is the subspace spanned by the vectors in π_α and π_β.

• Their *intersection:* $\pi_\alpha \cap \pi_\beta = \{x | x \in \pi_\alpha \text{ and } x \in \pi_\beta\}$ is the largest subspaces that belongs in both π_α and π_β.

Moreover, when we compare subspaces, we will need to measure "how far apart" two subspaces are. We measure the distance between two subspaces π_α and π_β as

$$d(\pi_\alpha, \pi_\beta) \triangleq \dim(\pi_\alpha + \pi_\beta) - \dim(\pi_\alpha \cap \pi_\beta). \quad (4)$$

It is also easy to see that

$$d(\pi_\alpha, \pi_\beta) = 2\dim(\pi_\alpha + \pi_\beta) - \dim(\pi_\alpha) - \dim(\pi_\beta). \quad (5)$$

Table 1 summarizes our notation. In our example over $\mathbb{F}_2^3$, if $\pi_\alpha = \pi_1 =< \mathbf{e}_1, \mathbf{e}_2 >$, and $\pi_\beta = \pi_2 =< \mathbf{e}_2 + \mathbf{e}_3, \mathbf{e}_1 >$, then $\pi_\alpha + \pi_\beta = \mathbb{F}_2^3$, $\pi_\alpha \cap \pi_\beta = \{\mathbf{e}_1\}$, and $d(\pi_\alpha, \pi_\beta) = 2$.

3. JOINT IDENTITY AND DATA CODING

We design codes that have the following properties:
1. **Identifiable codes:** Jointly convey identities and data.
2. **Error protection:** Can incorporate error correction.
3. **Scalability:** Scalable to the size of the network.
4. **Adaptability:** Adaptable to specific application needs.

We address each of these in turn, after describing our model.

3.1 Network Operation

We consider a sensor network with n sensor nodes where each node conveys to a data collection sink values from a small message set $\mathcal{M}_i$, together with its own identity. The information transfer comprises of the following basic tasks.

Source encoder: Each sensor i translates its data bits (or equivalently, information messages in $\mathcal{M}_i$) into a set of vectors in $\mathbb{F}_q^\ell$, using a pre-assigned codebook $\mathcal{C}_i$. As motivated in § 2.3 and amplified in § 2.4, the codebooks consist of d-dimensional subspaces[6] of $\mathbb{F}_q^\ell$, *i.e.*,

$$\mathcal{C}_i = \{\pi_j^{(i)} : 1 \le j \le |\mathcal{M}_i|\}, \quad i = 1, \ldots, n.$$

To transmit information to the sink, source i maps a measured value to one such subspace π and inserts in the network d vectors that span π.

Relay operation: In relaying information towards the sink, the sensor linearly combines all packets it has received (including that generated by itself) and transmits the combined packet to the next relays towards to the sink.

Data collector operation: Since each sensor node does the above encoding operation, the sink will observe vectors from the union of subspaces inserted by all the sources. In particular, if source i inserts the subspace π_i, the sink will observe vectors from the subspace $\pi_1 + \pi_2 + \ldots + \pi_n$. In all the codes we will describe here, the sink collects d packets and using the knowledge of the codebooks $\{\mathcal{C}_i\}$, decodes the sensor data along with its identity.

To be able to correctly decode at the sink, we need to ensure that every combination of sensor data results in a *distinct* union subspace. Such a condition would be sufficient for decoding and is called the *identifiable code* property.

DEFINITION 1 (**IDENTIFIABLE CODE**). *An identifiable code is a set of n codebooks $\mathcal{C}_i = \{\pi_j^{(i)} : 1 \le j \le |\mathcal{M}_i|\}$,*

[6]Although in principle we can use subspaces of different dimensions, we will for simplicity in most of this paper restrict our code design to subspaces of equal dimension, d-dimensional subspaces.

Table 1: Notation

$\mathbb{F}_q$	finite field with q elements
$\mathbb{F}_q^\ell$	vector space, where vectors have length ℓ and elements from $\mathbb{F}_q$
$< \mathbf{v}_1, \ldots, \mathbf{v}_n >= \{\mathbf{x} : \mathbf{x} = \sum_{i=1}^n \alpha_i \mathbf{v}_i\}$	vector space spanned or generated by the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$
$\pi \subseteq V$	π is a subspace of the vector space V
$\dim \pi$	dimension of subspace π
$\pi_\alpha + \pi_\beta = \{x + y \vert x \in \pi_\alpha, y \in \pi_\beta\}$	sum or union of two subspaces π_α, π_β
$\pi_\alpha \cap \pi_\beta = \{x \vert x \in \pi_\alpha \text{ and } x \in \pi_\beta\}$	intersection of two subspaces π_α, π_β
$d(\pi_\alpha, \pi_\beta) \triangleq \dim(\pi_\alpha + \pi_\beta) - \dim(\pi_\alpha \cap \pi_\beta)$	distance beween two subspaces π_α, π_β; see also (5)
π_α and π_β are distinct subspaces	$d(\pi_\alpha, \pi_\beta) \geq 1$
π_α and π_β are disjoint subspaces	$\pi_\alpha \cap \pi_\beta = \{0\}$
n	total number of sensor nodes
$\mathcal{M}_i$	set of data values for sensor i
$\mathcal{C}_i$	codebook for sensor i

Table 2: Coding for two sources

$\mathcal{C}_2/\mathcal{C}_1$	π_1	π_2	π_3
π_4	$\pi_1 + \pi_4$	$\pi_2 + \pi_4$	$\pi_3 + \pi_4$
π_5	$\pi_1 + \pi_5$	$\pi_2 + \pi_5$	$\pi_3 + \pi_5$
π_6	$\pi_1 + \pi_6$	$\pi_2 + \pi_6$	$\pi_3 + \pi_6$

$i = 1, \ldots, n$, with $\pi_j^{(i)} \subset \mathbb{F}_q^\ell$ such that we have $\pi_{i_1} + \ldots + \pi_{i_n} \neq \pi_{j_1} + \ldots + \pi_{j_n}$ when[7] $(i_1, \ldots, i_n) \neq (j_1, \ldots, j_n)$ and π_{i_k} is the subspace chosen by sensor k.

EXAMPLE 1. *Assume for simplicity we have two source nodes, S_1 using the codebook $\mathcal{C}_1 = \{\pi_1,\ \pi_2,\ \pi_3\}$, while S_2 the codebook $\mathcal{C}_2 = \{\pi_4,\ \pi_5,\ \pi_6\}$. Table 2 summarizes all outcomes. For this code to be identifiable, we want all 9 entries in Table 2 to correspond to distinct subspaces. For example, $\pi_1 + \pi_4$ should be a distinct subspace from $\pi_2 + \pi_5$.*

3.2 Identifiable Codes

We develop identifiable codes by decomposing the ℓ dimensional space $\mathbb{F}_q^\ell$ in disjoint subspaces, and restricting the codebook of each source inside one such disjoint subspace. We choose D-dimensional disjoint subspaces $\Pi_1, \ldots, \Pi_n$ such that their union is the whole space, *i.e.,* $\sum_{i=1}^n \Pi_i = \mathbb{F}_q^\ell$ and $nD = \ell$. Each sensor node i is assigned one such D-dimensional subspace Π_i. Then, the sensor maps each measurement data to one of $K_D(d)$ distinct subspaces of Π_i of dimension d (see also (3) for $K_D(d)$), where $d \leq D$.

Orthogonal Identifiable Codes

- Select n disjoint subspaces Π_i with $\sum_{i=1}^n \Pi_i = \mathbb{F}_q^\ell$
- Assign to sensor i the subspace Π_i
- Sensor i uses $\mathcal{C}_i = \{\pi_i \vert \pi_i \subset \Pi_i, \dim \pi_i = d\}$

[7]This inequality just means that there is at least one sensor k such that $i_k \neq j_k$.

We can think of this code construction as allocating D bit-positions of a packet to each source. The source employs only these D bits, and sends vectors that have zero value everywhere else apart these D positions. These D-bits span the disjoint D-dimensional space allocated to the source. For example, the source may use the $d = 1$ dimensional subspaces inside its space. Then it uses one of the $2^D - 1$ non-zero vectors (one-dimensional subspaces), and sends at each round one packet to the sink[8]. It is also easy to see that we can adapt our construction to give unequal number of bit-positions to different sensors in order to accommodate variations in measurement data sizes.

EXAMPLE 2. *In the example of Fig. 1 in § 2.2, we use binary vectors of length n and orthogonal spaces of dimensions $D = 1$. Then the codebook $\mathcal{C}_i$ can be described as $\mathcal{C}_i = \{< 0 >, < \mathbf{e}_i >\}$, where $\mathbf{e}_i$ is the unit vector with 1 in position i.*

As we discussed, our code construction effectively divides the packets into portions allocated to each sensor. However, this is just one approach, and there exist various other methods to share the vector space among the sources, that rely more on algebraic properties. As we will see, a more sophisticated approach might be needed for constructing identifiable codes that are scalable and adaptable (§3.4 and §3.5).

3.3 Error Correction

As mentioned in Section 3.1, packets get dropped due to link congestion or error. In this case, to be able to still deliver the sensor data and identity, we need to add error protection to identifiable codes. One method is to rely on MAC

[8]One might argue that our construction is slightly inefficient as we are not using the all-zero vector (that corresponds to a zero-dimensional subspace). However, we implicitly use the all-zero vector to convey the fact that a sensor is not active during a particular round. This allows us to differentiate between "sensor not active" and "sensor observes value zero". We can also use the all-zero vector, by incorporating the trivial subspace $< 0 >$ in the codebook.

layer retransmissions to provide resilience to errors. Here we present an alternative forward error correction approach, that does not require feedback. This approach is well matched for the cases where feedback cannot readily be used[9], or when sensors fail, and their lack of transmission would have the same effect as erasures.

Our code construction builds on the orthogonal identifiable code construction, where we allocate one of the D-dimensional disjoint subspaces $\Pi_1, \ldots, \Pi_n$ to each sensor. We want to now construct each codebook $\mathcal{C}_i$ to *no longer contain all* $K_D(d)$ distinct d-dimensional subspaces inside Π_i, but instead, a set of d-dimensional subspaces that are "far apart". Indeed, to protect against errors, we need to introduce "redundancy" into identifiable codebooks, by separating the subspaces chosen in the codebooks by a certain "distance"[10]. To this end, the distance measure between subspaces defined in (4) is useful. We define the minimum distance of the codebook $\mathcal{C}_i$ as the closest two subspaces in the codebook can get. More formally,

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

where $d(\pi_\alpha, \pi_\beta)$ was defined in (4).

Next, we investigate the erasure correction capabilities of a code with a given distance. We restrict our attention to the bit-positions inside the packet that the transmitted subspace sensor i uses. The effect of erasures is to eliminate some of the "dimensions" from the transmitted subspace, making the received vectors to lie in a lower dimensional sub-subspace of the transmitted subspace. That is, the sink, instead of receiving d packets which at the bit-positions corresponding to node i have d linearly independent vectors, it will receive instead $d - r_i$ such packets, where r_i are the erasures node i experienced. If the codebook contains two subspaces which have a common $(d-r_i)$-dimensional subspace (*i.e.,* intersect in $(d - r_i)$ dimensions), they could be potentially indistinguishable. Making this intuition formal, Theorem 1 determines how many erasures our identifiable codes can provably correct. A more general form of this theorem is proved in Appendix A.

THEOREM 1. *Consider a set of codebooks $\mathcal{C}_i$ used over a channel that erases r_i packets from source i. If*

$$2r_i < D(\mathcal{C}_i), \tag{7}$$

where $D(\mathcal{C}_i)$, defined in (6), is the minimum distance of the codebook $\mathcal{C}_i$, then we can recover the messages for source i.

[9]This is also useful when we utilize broadcasting (see §4.1.2).

[10]Also note that traditional erasure correcting codes (like the Reed-Solomon code [17]) would not work in our case since the code would *not* be oblivious to linear mixing packets emanating from the *same* source.

Table 3: Code with five codewords, where each codeword is a 2-dimensional subspace of a 4-dimensional space.

$\pi_1 =$	$< [0\ 1\ 1\ 0],\ [1\ 0\ 0\ 1] >$
$\pi_2 =$	$< [0\ 0\ 1\ 0],\ [0\ 1\ 0\ 1] >$
$\pi_3 =$	$< [1\ 1\ 1\ 0],\ [1\ 1\ 0\ 1] >$
$\pi_4 =$	$< [1\ 0\ 0\ 0],\ [0\ 1\ 0\ 0] >$
$\pi_5 =$	$< [1\ 0\ 1\ 0],\ [0\ 0\ 0\ 1] >$

Orthogonal Identifiable Code for r Erasures

- Select n disjoint subspaces Π_i with $\sum_{i=1}^{n} \Pi_i = \mathbb{F}_q^\ell$
- Assign to sensor i the subspace Π_i
- $C_i = \{\pi_i | \pi_i \subset \Pi_i, \dim \pi_i = d\}$ with $D(C_i) > 2r$

The final question is to ask how one would decode an error correcting identifiable code. Since the only thing preserved are the subspaces, a natural choice would be to choose the "allowable" subspace (which is part of the codebook) that is closest to the received subspace. For example, if the codebook contains the subspaces $\mathcal{C}_i = \{\pi_j\}$ and the sink (always only considering the bit-positions that sensor i uses) receives a subspace π_R, then the sink will decode $\hat{\pi}_i = \arg\min_{\pi_i} d(\pi_i, \pi_R)$. Note that to calculate $d(\pi_i, \pi_R)$ we can create a matrix $\mathbf{R}$ that has $d + d - r_i$ rows, with the first d rows being any d basis vectors of π_i and the remaining $d - r_i$ rows being the $d - r_i$ received vectors that span π_R. Then we can calculate the rank of the matrix $\mathbf{R}$ to find $\dim(\pi_i + \pi_R)$ and substitute this in (5). All the previous ideas are perhaps better illustrated through an example.

EXAMPLE 3. *Assume we allocate to each source $D = 4$ bits. Within this space, each source can employ the code $\mathcal{C}$ with 5 codewords described in Table 3. Each codeword of $\mathcal{C}$ is a 2-dimensional subspace. The distance between any two codewords equals four, and thus $D(\mathcal{C}) = 4$.*

The source, to convey for example the first codeword $\pi_1 =< [0\ 1\ 1\ 0], [1\ 0\ 0\ 1] >$, can insert in the network two packets, one containing the vector $[0\ 1\ 1\ 0]$, and the other the vector $[1\ 0\ 0\ 1]$. Because the codewords consist of non-intersecting subspaces, if the sink receives any one vector *in the subspace π_1, it will uniquely identify this subspace. For example, the receiver may observe the vector $\mathbf{y} = [1\ 1\ 1\ 1]$. To decode, the receiver needs to check the rank of five matrices, e.g.,*

$$\textit{rank} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} = 2, \textit{ while rank} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} = 3.$$

Thus $\mathbf{y}$ belongs in π_1 and not in π_2.

3.4 Scalability

In the identifiable code design advocated in Section 3.2, we see that for n sources, each using D-dimensions, the vector length is nD. For scalability, we would like to make this

more efficient when n grows.

To make our code construction scalable, we will consider the case where at most m vectors get combined. This assumption is motivated through three observations:
(i) The identifiable codes designed in Section 3.2 allowed us to decode the identity and data even if *all* the sources' vectors get mixed. However, vectors get combined only if their paths to the sink overlap. For the example in Figure 2 the sink has a single neighbor node B and all source vectors get combined. If the sink has k neighbor nodes with the sources symmetrically deployed along these k directions, we can then think of Fig. 2 as the "slice" of the network corresponding to one of these neighbors, and each received packet at the sink will contain approximately $\frac{n}{k}$ combined vectors.
(ii) Not all sensors may be actively measuring during every round. For example, when sensing anomalies, we expect a small fraction of all potential sources to send information.
(iii) We can artificially restrict the number of sources that get combined, by appending to each packet a few bits to count the number of combined packets it contains.

Our problem statement can now be summarized as follows. Given n sources, the sink is going to observe packets that contain linear combinations of *at most* m sources. We want to design codes that allow us, by receiving each combined vector, to determine which is this subset of combined sources, and what are their transmitted messages.

For simplicity we describe our code construction for the case where each sensor either transmits a single value or remains silent (to indicate it has perceived a certain event or not), but the design can be easily extended to a larger measurement set. Before delving into the code construction, we emphasize that our code utilizes vectors of length ℓ that is much smaller than the number of sensor nodes n. For example, Fig. 5 plots upper and lower bound on ℓ for the case where $m = 2$ and where $m = 20$ vectors get combined.

Our code construction proceeds as follows. Select a linear code of length n, minimum distance $2m+1$, and redundancy ℓ, with ℓ as small as possible. Consider the $\ell \times n$ parity check matrix $\mathbb{H}$. Assign to each source a codebook with the 1-dimensional subspace spanned by one column of $\mathbb{H}$.

Scalable Codes for m Combined Packets
- Let $\mathbb{H}$ be the $\ell \times n$ parity check matrix of a binary code with minimum distance $2m+1$
- Then $C_i = \{< h_i >\}$ where h_i is a column of $\mathbb{H}$

The reason this construction satisfies our design goals, stems from a well known property the columns of matrix $\mathbb{H}$ satisfy. Let $\mathcal{A}$ denote the set of these vectors. Then, if the code has minimum distance $2m+1$, *any set of* $2m$ *vectors in* $\mathcal{A}$ *are linearly independent* [17]. This directly implies the following properties. For vectors in $\mathcal{A}$:
(P1) Any set of $0 < \beta < 2m$ vectors span a distinct β-dimensional subspace.

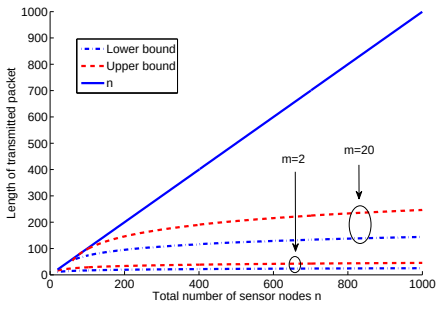


Figure 5: Bounds on the length ℓ of transmitted vectors when $m = 2$ and $m = 20$ sources get combined, as a function of the number of sensor nodes n.

(P2) The distance between two subspaces π_1 and π_2, where each subspace is spanned by a different set of $0 < \beta < 2m$ vectors, equals $\min\{\beta, 2m - \beta\}$.
From property *(P1)* we see that the sink receives distinct subspace for *every* distinct set of m sources, and therefore is able to decode the identities and the information. Thus, even though there are $n = |\mathcal{A}|$ possible sources, we do not need to use vectors of length proportional to n, but instead, of length $\ell \geq 2m$.

In order to quantify our savings, we need to examine how ℓ scales with m. This is related to a well-studied problem in coding theory, namely, for a given code length $n \triangleq |\mathcal{A}|$, and a given minimum distance $2m+1$, what are upper and lower bounds on the number of codewords $A(n, m)$ this code can have [17]. Using the Gilbert-Varshamov lower bound and the sphere packing upper bound [17], it can be shown that for large values of n,

$$nH_2(\frac{2m+1}{2n}) \leq \ell \leq nH_2(\frac{2m+1}{n}), \qquad (8)$$

where H_2 is the binary entropy function, namely, $H_2(p) = -p\log p - (1-p)\log(1-p)$. Fig. 5 plots the bounds in (8) as a function of n, for $m = 2$ and for $m = 20$. We conclude that our proposed code design results in using a fraction of the length n, that goes to zero as the ratio $\frac{2m+1}{n}$ goes to zero.

EXAMPLE 4. *Using a table of the best codes known [17] we can see for example that, there exist binary linear codes of length* $n = 512$ *with redundancy* $\ell = 18$ *and minimum distance* $2m + 1 = 5$. *Thus in a sensor network with* 512 *nodes if at most* $m = 2$ *source vectors get combined, we need to use vectors of length* $\ell = 18$.

We finally note that to incorporate forward error correction, we can modify our code construction by separating the subspaces akin to the ideas in § 3.3.

3.5 Adaptability

The specific applications for which a sensor network could be deployed might impose a diverse set of requirements for data collection. In this section we explore how our codes

Table 4: Code that checks consistency

$\mathcal{C}_2/\mathcal{C}_1$	π_1	π_2	π_3
π_4	α	β	γ
π_5	ϵ	α	δ
π_6	ζ	θ	α

can be easily adapted to suit such requirements. Through examples, we develop a sense of the flexibility offered by our network operation architecture, which is the property we want to highlight.

We consider two different application requirements. (i) *Data dependent identifiability:* We need to know about measurement information and identities when certain combinations of sensor data occur. (ii) *Mixing statistics and identification:* Computing functions of sensor measurements when data falls in a certain set.

Note that the data collector can send a signal to the sensors to change their operating mode to suit the requirements above, *without* changing the relaying (mixing) operation at the intermediate nodes. This would just make the sensors operate with a different codebook to map its observations to the transmitted packet. This flexible end-to-end change of operation is a distinct advantage of our architecture.

Data dependent identifiability. Suppose that we are interested in identifying the sensors only when certain combinations of measurement data occurs. For example, we care to know the sensor identities and measurements when there are discrepancies in their observations.

EXAMPLE 5. *Consider two source sensors each observing one of three possible values in the set $\mathcal{M} = \{0, 1, 2\}$. Source S_1 employs the codebook $\mathcal{C}_1 = \{\pi_1,\ \pi_2,\ \pi_3\}$, while source S_2 the codebook $\mathcal{C}_2 = \{\pi_4,\ \pi_5,\ \pi_6\}$.*

If we are interested in learning about identities only when sensor measurements do not match, then, the function described by Table 4, needs to be implemented. Note that when the two sensors observe the same value, no matter what this value is, the sink receives the same subspace. To implement this function we use vectors of size $\ell = 2d$, and codewords d-dimensional subspaces of $\mathbb{F}_2^\ell$. Given any d-dimensional subspace $\pi \subset \mathbb{F}_2^\ell$, by definition its complement $\bar{\pi}$ is also a d-dimensional subspace that satisfies $\pi + \bar{\pi} = \mathbb{F}_2^\ell$. Select the codebook $\mathcal{C}_1$ to contain three distinct *d-dimensional subspaces $\{\pi_1,\ \pi_2,\ \pi_3\}$ of $\mathbb{F}_2^\ell$. We construct $\mathcal{C}_2$ by using $\pi_4 = \bar{\pi}_1$, $\pi_5 = \bar{\pi}_2$, and $\pi_6 = \bar{\pi}_3$. Note that the sink receives a vector in the space $\pi_{i,j} = \pi_i + \bar{\pi}_j$, when i, j are respectively the measurements in sensor S_1, S_2. We will then get the desired property, since $\pi_{i,i} = \mathbb{F}_2^\ell$ and by construction $\pi_i + \bar{\pi}_j$ is distinct for $i \neq j$. Note that each individual node does not need to know what the other node has observed. The coding scheme, in a distributed manner, ensures the desired property.*

This example illustrates how we can adapt the code to suit data dependent identifiability requirements. Another example could be that of sensor clustering where we want to coarsely characterize the spatial field into regions of interest. By assigning to all the sensors in each group the same codebook, we can know about groups with interesting measurements, which can be examined more closely later through *multi-stage collection*: by asking the subset of sensors to switch to identifiable codebooks[11].

Mixing statistics and identification. In many current applications of sensor networks, only statistics (like average, maximum, histograms) are needed and not the sensor identities. However, one can envisage a situation where we need to collect statistics when the sensor data is in a certain state and want to obtain identity/data in other states. This needs to be done oblivious to the intermediate node operations. We can adapt our code to satisfy this requirement as seen in the Example 6.

EXAMPLE 6. *Consider an application where the sensor measurement set $\mathcal{M}$ is partitioned into two parts: $\mathcal{M} = \mathcal{M}^I \cup \mathcal{M}^H$. Identity of a node is important for values in $\mathcal{M}^I$ and only statistics (histogram) of nodes observing values in $\mathcal{M}^H$ is needed. We use vectors that consist of two parts: the first part P^H collects the histogram statistics for the $\mathcal{M}^H$ values, while the second part P^I implements one of our coding schemes for the $\mathcal{M}^I$ values. The interesting point is that, we can design our schemes so that intermediate nodes perform exactly the same operation on both parts P^H and P^I of the vector, and are oblivious to this distinction. To do this, we use operations over a higher field $\mathbb{F}_q$ that has size q of order n, where n is the number of sensors. We can then use $|\mathcal{M}^H|$ symbols over $\mathbb{F}_q$ to construct P^H, where the symbol in position i will keep track of how many nodes have observed the corresponding value in $\mathcal{M}^H$. Thus addition of the P^H parts of the packet will create the desired histogram; and we can design the part P^I, again over $\mathbb{F}_q$, to allow us to convey the identity of up to m nodes that observed a given value.*

Other adaptations. Up to now, we have examined sensor networks that employ a single sink, and that create a tree topology to connect the sources to the sink. However, sensor networks with a large number of nodes may have more than one sinks; moreover, our topology may not necessarily be a tree[12]. All our code designs are oblivious to the network structure and the number of sinks; indeed, this is one of the strengths of our design.

4. DATA COLLECTION PROTOCOL

[11]We can have further efficiency by using scalable codebooks as described in Section 3.4.

[12]See the broadcasting collection protocol in §4.1.2.

In this section we explain some of the systems issues we encountered when implementing identifiable codes for energy-efficient data collection. First, the codes are centrally computed and distributed to the sensors through dissemination.Each sensor then enters a loop of "reporting rounds," where it produces one measurement per round; nodes do not run any initial synchronization protocol, *i.e.,* each node starts its first round independently from every other. Each measurement is jointly encoded with the sensor's identity to produce d vectors of size ℓ, where d is the dimension of the subspaces in the codebooks and ℓ the employed vector length. We now describe how, and at what cost, the network relays this information to the sink.

4.1 Relaying

Each node maintains d buffers of length ℓ, which store the vectors corresponding to its latest measurement. Whenever the node receives a packet from a neighbor, it XORs its contents to one of the buffers. Occasionally (at least once per round), the node packs the contents of each buffer into a separate packet and forwards the d packets towards the sink. The question is *when* and *where* to forward and how to deal with link-layer losses and collisions; depending on network characteristics, we follow one of two protocols.

4.1.1 Routing with Retransmissions

The nodes build and maintain a spanning tree rooted at the sink, as in the Collection Tree Protocol (CTP) [28]. In each round, an individual node tries to collect information from all its children, *then* forwards everything to its parent, so that each node transmits exactly d packets. Losses and collisions are handled through link-layer acknowledgments and retransmissions.

More specifically, in the beginning of each round, each node starts a "transmission timer"; it forwards the contents of its buffers to its next hop as soon as (i) it has received packets from all its children, and/or (ii) the transmission timer expires and the buffer is non empty; packets received after the timer expires are immediately forwarded to the next hop. As a result, as long as each node correctly determines the number of children it should wait for, each node transmits exactly d packets per round. Of course, when the tree topology changes, a node may unexpectedly hear from (new or delayed) children *after* its transmission deadline, in which case it has to transmit more than d packets.

The advantage of this approach is simplicity—all a node has to do is count its children. The disadvantage is that efficiency is affected by link-layer errors (losses and collisions) in two ways: First, errors lead to retransmissions, increasing the number of data/acknowledgment frames and duplications. Second, they lead to frequent changes in the tree topology, causing nodes to send and receive unexpected packets. Hence, this strategy is suited to low-noise, stable networks.

4.1.2 Broadcasting

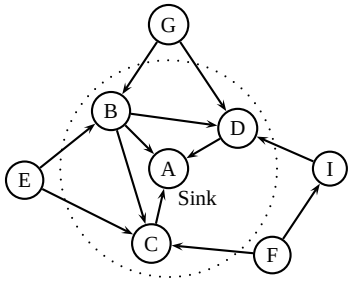


Figure 6: Nodes divided into two sets (inside and outside the circle) according to their distance from the sink.

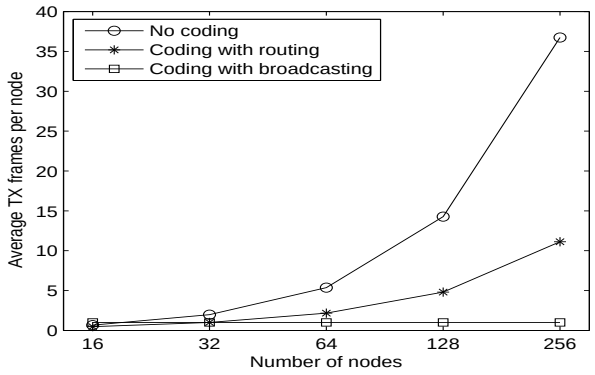
Nodes arrange themselves in different "strides" around the sink, such that nodes in stride i are better connected to the sink than nodes in stride $i+1$. Each node broadcasts the contents of each buffer exactly once per round, such that the nodes in stride $i+1$ broadcast before the nodes in stride i—*i.e.,* information propagates from outer to inner strides similar to how it propagates from children to parents in Section 4.1.1. There are no acknowledgments or retransmissions: to overcome link-layer errors, the protocol relies on (i) its multi-path nature (at least one neighbor is likely to hear each node's transmission) and (ii) the error-correcting capabilities of the codes.

More specifically, each node schedules its next-round broadcasts based on the last broadcast it hears from a neighbor in the current round: if it hears the last neighbor broadcast at time t, it schedules its own next-round broadcasts in the interval $(t + T_R - T_S, t + T_R)$, where T_R is round duration and T_S is a "stride slot" on the order of a few seconds. The idea is best illustrated through the example of Figure 6: in the first round, the sink A broadcasts a beacon; node B hears it and schedules its next-round broadcasts within one stride-slot before A's, i.e., B arranges itself in stride 1; in the second round, node E hears B's broadcast and schedules its next-round broadcasts within one stride-slot before B's, *i.e.*, E arranges itself in stride 2. Note that a node does not need to know which specific stride it is in, it just schedules its next-round broadcasts relative to the last neighbor it hears.

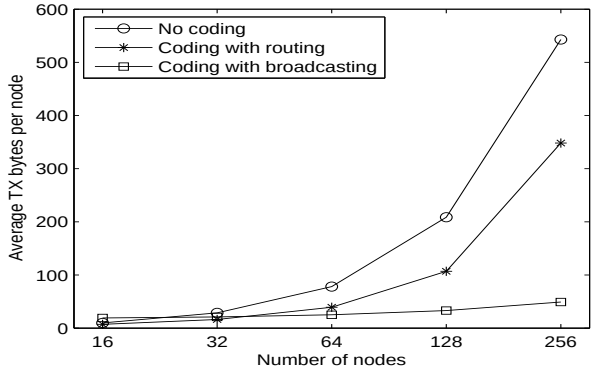
The efficiency of this approach is not affected by link-layer errors: each node always transmits exactly d packets per round, independently from network conditions and without the need to collect and maintain valid routing information; this makes it ideal for weakly connected networks with severe link-layer errors and frequent sensor failures. The fixed number of per-round transmissions is precisely what allows us to benefit from the multi-path nature of broadcasting *and* be energy efficient at the same time—a conventional flooding protocol where nodes simply broadcast all incoming packets would be anything but energy efficient.

4.2 Memory and Complexity

We implemented both collection protocols as TinyOS [27] applications for TinyNode [30] motes. In terms of memory, each node needs d buffers of size ℓ, where d depends on the amount of error correction, while ℓ depends on the num-



(a) Average per-node frame transmissions



(b) Average per-node byte transmissions

Figure 7: Average per-node frame and byte transmissions per round as a function of the number of sensors. The sensors are randomly placed on a square topology, 1 sensor per $100m^2$. The channel characteristics are of an indoors environment with low noise.

ber of (mixed) sensors; for practical codes, d varies from 1 to 3, while ℓ is on the order of a few tens of bytes—we don't recommend longer than 16-byte vectors. Moreover, each node needs a small (< 8 bits) counter per neighbor (to store the number of packets received from each neighbor in this round) and a timer for scheduling its transmissions. In terms of operational complexity, each node performs transmissions, receptions, and binary XOR combinations. In the case of routing with retransmissions, it also runs CTP's distance-vector protocol for building and maintaining the spanning tree, which consists of reading a 2-byte routing gradient from the headers of incoming packets, comparing it to the local gradient, and updating the latter accordingly.

5. EXPERIMENTAL EVALUATION

5.1 Metrics and Setup

Our goal, for sensor networks, is to reduce the amount of traffic needed to successfully communicate measurements to the sink. Hence, our experiments compare data-collection protocols in terms of (i) the amount of traffic (frames and bytes) they generate, and, (ii) in scenarios with end-to-end

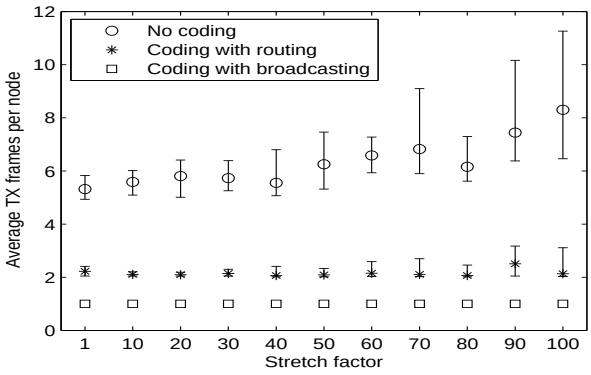


Figure 8: Average per-node frame transmissions per round as a function of topology shape (the x-axis represents the ratio between the two dimensions of the topology): The sensors are randomly placed, 1 sensor per $30m^2$ (we chose this as the lowest density for which conventional data collection does not suffer end-to-end message loss throughout the experiment). The channel characteristics are similar with Figure 7.

loss, the percentage of sensors that successfully communicate their message to the sink.

Our baseline is a tree-based collection protocol that does not perform any joint identity-message coding (we refer to this as "conventional data collection"): The nodes use CTP [28] to build and maintain a spanning tree rooted at the sink. In each round, each node sends out a packet that contains its identity and measurement, specified in separate fields; whenever a node receives a packet from a neighbor, it immediately forwards it to its next hop. Link-layer errors are handled through acknowledgments and retransmissions.

For consistency, all the results presented concern a simple application, where the sensors periodically communicate their identities and a single-bit message to the sink (we also ran experiments with 2- to 4-bit messages, but the results did not vary significantly). For each experiment, we choose an appropriate joint identity-message code, combine it with our coding-with-routing (§4.1.1) and coding-with-broadcasting (§4.1.2) protocols, and compare the outcome to the conventional data collection.

We used the TOSSIM simulator [16] to test these three protocols under different scenarios. Each scenario corresponds to a different set of network characteristics—number of nodes, sensor density, topology shape, and level of noise. For each scenario, we use the corresponding characteristics to generate 10 different topologies, then run the protocol under test for 10 rounds in each topology; hence, each data point in our graphs represents an average over 100 rounds. When the variance of the represented value over the 100 rounds is significant (Figs 8, 9(a), and 10(a)), instead of the average, we plot its minimum-maximum range and median.

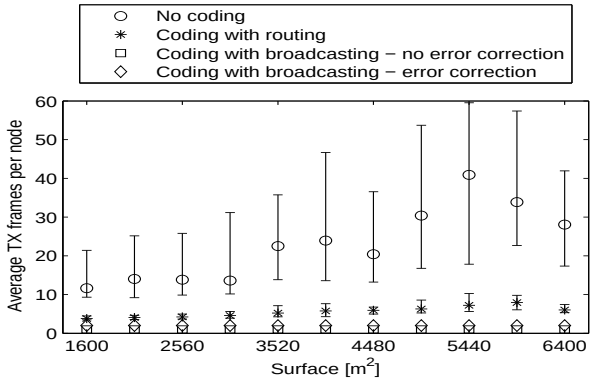
5.2 Efficiency

We first look at well connected networks with infrequent link-layer errors—an indoors setting with low noise and one

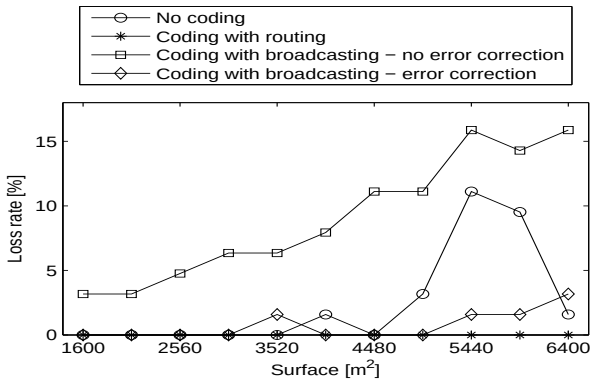
sensor every 30 to 100 square meters (depending on the experiment). Hence, we use a one-dimensional identifiable code without error correction (§3.2), where each node maps its message to a single vector as in Example 2.

We run two sets of experiments: first, we fix the shape of the topology (to a square), and vary the number of sensors; then we fix the number of sensors (to 64), and vary the shape of the topology, starting from a square and increasing the ratio between its two dimensions, until we get a narrow corridor. In both cases, the goal is to gradually increase the average distance of the sensors to the sink and show how that affects the behavior of the protocols.

Figures 7 and 8 show the results: Compared to conventional data collection, both our protocols reduce the total number of frames and bytes generated in each round by a factor of 2 to 4; the benefit increases with the number of nodes and as the topology becomes longer and narrower. This is because, in conventional data collection, intermediate nodes simply forward packets; hence, the larger the average distance to the sink, the larger the number of generated frames (and bytes). On the contrary, in our protocols, each intermediate node combines the packets sent by its children or neighbors into a single packet; hence, the number of per-node transmissions remains at 1 (in coding with broadcasting) or close to 2 (in coding with routing, which uses acknowledgments), even as the average distance to the sink increases.



(a) Average per-node frame transmissions



(b) End-to-end message loss

Figure 9: Efficiency and loss as a function of topology surface: The sensors are randomly placed. The channel characteristics are of an indoors environment with very high noise (the trace from the Stanford Meyer library included with the TinyOS distribution).

5.3 Error Resilience

Next, we consider a noisy setting with frequent link-layer errors. We use two different codes: (i) a one-dimensional code with no error correction as above, and (ii) a two-dimensional code with single-error correction (§3.3), where each node maps its message to two vectors, and the sink can decode the message as long as it receives one of them. We try four data-collection methods: (i) conventional data collection, (ii) coding with routing (without error correction), (iii) coding with broadcasting with, and (iv) without error correction.

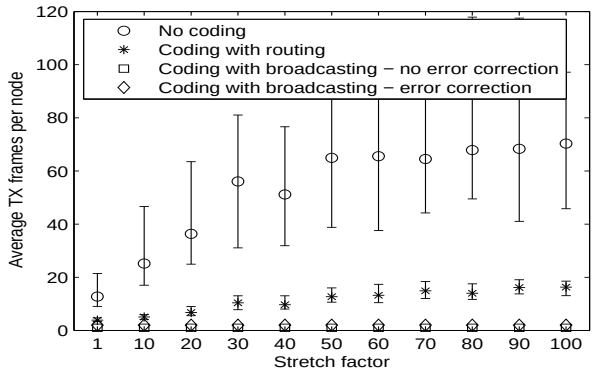
First, we run a set of experiments, where we fix the number of sensors (to 64) and gradually increase the surface of the covered area, *i.e.,* decrease the sensor density. The goal is to show how the behavior of the protocols changes with link-layer quality.

Fig. 9(a) shows the number of frames generated by each protocol (we do not show the number of generated bytes, as it follows the same curve): Again, compared to conventional data collection, our protocols reduce the total number of frames and bytes generated in each round by several factors. An interesting difference from the low-noise setting is that, as sensor density decreases, coding with routing becomes slightly less efficient; this is due both to link-layer retransmissions, as well as frequent changes in the tree topology that cause the nodes to make wrong guesses about how many packets they should combine (§4.1.1).

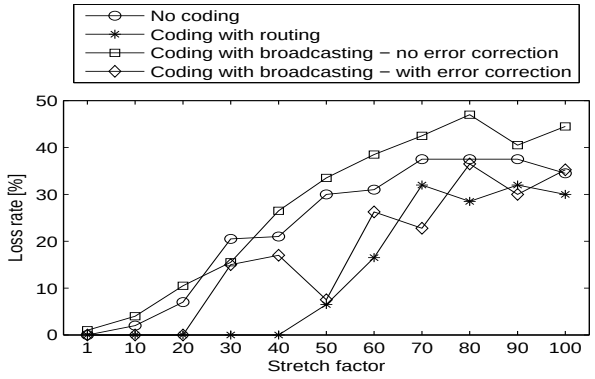
Coding with broadcasting, on the other hand, generates the same amount of traffic independent of link-layer quality—because it uses no acknowledgments/retransmissions and forces each node to transmit a fixed number of times per round. Of course, these two features that make it efficient can also cause end-to-end message loss. Indeed, as shown in Figure 9(b), coding with broadcasting and no error correction suffers more loss than any other method, especially in larger topologies, where nodes are less likely to be heard by their neighbors; error correction fixes this problem at the cost of a second transmission per node in each round.

We also run a set of "topology stretching" experiments as in §5.2, *i.e.,* fix the number of sensors and vary the shape of the topology from a square to a narrow corridor. The goal is to create increasingly longer paths and more contention close to the sink, and show how that affects the behavior of the protocols.

Figures 10(a) and 10(b) show, respectively, the amount of generated traffic and end-to-end message loss. The trends remain the same: conventional data collection generates the most traffic, coding with broadcasting and no error correction generates the least—but suffers the biggest end-to-end message loss, and broadcasting with error correction strikes



(a) Average per-node frame transmissions



(b) End-to-end message loss

Figure 10: Efficiency and loss as a function of topology shape (the x-axis represents the ratio between the two dimensions of the topology): The channel characteristics are similar with Figure 9.

the best balance between efficiency and end-to-end connectivity. We should note that some of the topologies used in this last experiment are somewhat extreme, which explains the high levels of loss incurred by all four protocols. For instance, the topology that corresponds to stretch factor 100 is a $400 \times 4m^2$ corridor with heavy noise; we do not suggest doing data collection in such an environment with 64 nodes and a single sink—we are only showing these results to give a sense of the protocols' limits.

5.4 Scalability

Finally, we demonstrate the performance of scalable identifiable codes developed in §3.4. We consider a network of 256 nodes and three different codes: (i) a scalable $\ell = 16$-bit code that allows combinations of $m = 2$ sources, (ii) a scalable $\ell = 23$-bit code that allows combinations of $m = 3$ sources, and (iii) an identifiable 32-byte code that allows arbitrary combinations of sources. We combine these codes with our coding-with-routing protocol and compare the results against conventional data collection. Figure 11 shows the trade-off between packet size and overall efficiency.

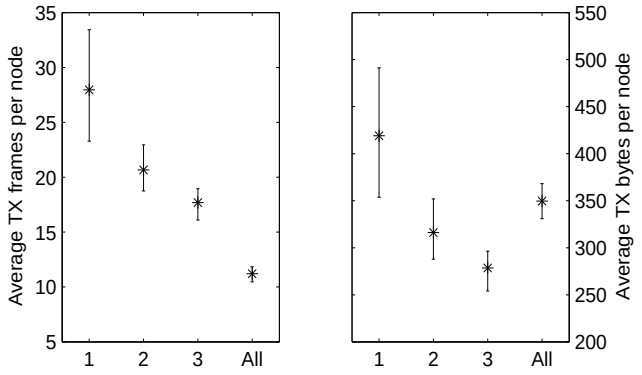


Figure 11: Trade-off between packet size and overall efficiency. The x-axis represents the maximum number of combined packets—1 corresponds to conventional data collection.

Recently, techniques inspired from coding and network coding have been successfully used to harness the broadcasting capabilities of the wireless medium [25, 13, 14, 6, 26, 23], to implement intelligent in-network storage [12], and to provide resilience in lossy environments [10, 9]. These coding techniques do not offer a viable solution for the identity-aware sensing. Indeed, traditional network coding requires appending to each packet a number of training bits proportional to the number of source packets, which would have prohibitive communication overhead in our case [3, 7]. These training bits are necessary for the sink to learn the overall transfer function[13] and thus be able to decode. An essential motivation for using subspace codes in our work is that they do not require this knowledge and thus dispense with the use of training bits.

As far as we know, our work is the first to develop subspace codes for sensor networks. Subspace codes have been studied in the context of non coherent communication over fading point-to-point wireless channels [19]. Such codes have also been advocated for quantum communication [2], and were also recently proposed for use over networks that employ network coding, to provide error and erasure correction [15]. All the previous constructions are addressed to a single source and thus do not encode the source identity. Moreover, they are designed for large packet information transfer. In our work, we develop constructions that incorporate in the code the identity of multiple sources with low communication overhead.

Significant research effort has been invested in reducing the communication overhead, by using distributed data aggregation techniques inside the sensor network. These techniques exploit data correlation to perform compression [21, 22, 18], or, calculate functions of the observed measure-

[13]Transfer function is the linear operations applied by the network to the source packets, summarized by $\mathbf{G}_i$ in (1). Note that the transfer function changes from round to round, due to topology changes, different synchronization between nodes, and different operations nodes perform.

6. RELATED WORK

ments, such as the average value [18, 1]. None of these is applicable in the case where we need to convey node identities. A naive aggregation approach would be to package at each node the identities and information bits received in a single packet. This would result in unequal length packets of a size that increases prohibitively as we approach the sink, and requires sophisticated in network content processing. Also, unlike conventional aggregation techniques, our approach does not suffer from irregular spatial sampling [8], as our protocol is oblivious to the network structure.

7. CONCLUSIONS AND OUTLOOK

We formulate the problem of identity aware sensor networks to capture applications where, as illustrated in §2.1, identity of the sensor as well as the measurement is of importance. We propose a data collection protocol based on combining identity and data through a subspace code. As far as we know, this is the first such approach. We demonstrate its effectiveness both through analysis and implementation on a TinyOS sensor platform, for energy efficiency benefits, load balancing, loss resilience and scalability.

8. REFERENCES

[1] A. Boulis, S. Ganeriwal, and M. B. Srivastava, "Aggregation in sensor networks: An energy-accuracy tradeoff", *IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, pp. 128–138, Mar. 2003.

[2] 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*, vol. 44, no. 4, pp. 1369–1387, Jul. 1998.

[3] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding", *Allerton Conference on Communication, Control, and Computing*, 2003.

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

[5] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model driven data acquisition in sensor networks", *Conference on Very Large Data Bases (VLDB)*, pp. 588–599, Toronto, Canada, 2004.

[6] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "A network coding approach to energy efficient broadcasting: From theory to practice", *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1–11, Barcelona, Spain, Apr. 2006.

[7] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "Network coding: An instant primer", *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 63–68, Jan. 2006.

[8] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin, "Coping with irregular spatio-temporal sampling in sensor networks", *Workshop on Hot Topics in Networks (HotNets)*, 2003.

[9] M. Ghaderi, D. Towsley, and J. Kurose, "Network coding performance for reliable multicast", *Military Communication Conference (Milcom)*, 2007.

[10] Z. Guo, P. Xie, J.-H. Cui, and B. Wang, "On applying network coding to underwater sensor networks", *1st ACM International Workshop on Underwater Networks*, pp. 109–112, California, USA, Sep. 2006.

[11] Horn and Johnson, "Matrix Analysis", Cambridge University Press, 1990.

[12] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth codes: maximizing sensor network data persistence", *ACM SIGCOMM*, vol. 36, 2006.

[13] S. Katti, S. Gollakota, and D. Katabi, "Embracing wireless interference: analog network coding", *ACM SIGCOMM*, pp. 397–408, Kyoto, Japan, Aug. 2007.

[14] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding", *ACM SIGCOMM*, 2006.

[15] R. Koetter and F. Kschischang, "Coding for errors and erasures in network coding", *IEEE International Symposium on Information Theory (ISIT)*, 2007.

[16] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications", *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 126–137, Los Angeles, 2003.

[17] F. J. MacWilliams and N. J. A. Sloane, "The theory of error correcting codes", North-Holland Mathematical Library, 1983.

[18] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks", *Symposium on Operating Systems Design and Implementation (OSDI)*, vol. 36, pp. 131–146, Boston, MA, 2002.

[19] F. Oggier, N. J. A. Sloane, S. N. Diggavi, and A. R. Calderbank, "Non-intersecting subspaces with finite alphabet", *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4320–4325, Dec. 2005.

[20] G. Pottie and W. Kaiser, "Principles of embedded networked systems, *Cambridge University Press*, 2005.

[21] S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense sensor network", *IEEE Signal Processing Magasine*, vol. 19, no. 2, pp. 51–60, Mar. 2002.

[22] A. Scaglione and S. D. Servetto, "On the interdependence of routing and data compression in multi-hop sensor networks", *Wireless Networks*, vol. 11, no. (1-2), pp. 149–160, 2005.

[23] S. Sengupta, S. Rayanchu, and S. Banerjee, "An Analysis of Wireless Network Coding for Unicast Sessions: The Case for Coding-Aware Routing", *IEEE Conference on Computer Communications (Infocom)*, Anchorage, Apr. 2007.

[24] D. Tulone and S. Madden, "PAQ: Time series forecasting for approximate query answering in sensor networks", *3rd European Workshop on Wireless Sensor Networks*, pp. 21–37, Zurich, Switzerland, Feb. 2006.

[25] Y. Wu, P. A. Chou, and S. Y. Kung, "Minimum-energy multicast in mobile ad-hoc networks using network coding", *IEEE Transaction on Communications*, vol. 53, no. 11, pp. 1906–1918, Nov. 2005.

[26] S. Zhang, S. Liew, and P. Lam, "Physical layer network coding", *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 24–29, Los Angeles, USA, Sep. 2006.

[27] "TinyOs", http://www.tinyos.net/.

[28] "The Collection Tree Protocol (CTP)", http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html.

[29] "Dissemination", http://www.tinyos.net/tinyos-2.x/doc/html/tep118.html.

[30] "TinyNode", http://www.tinynode.com/.

Appendix A

The received subspace π_R at the sink node can be expressed as [15]

$$\pi_R = [\mathcal{H}_{k_1}(\pi_1) + \cdots + \mathcal{H}_{k_N}(\pi_n)] \oplus \pi_E, \tag{9}$$

where π_i is the transmitted subspace by source S_i, π_E is an error subspace summarizing additive errors, and $\mathcal{H}_{k_i}$ are erasure operators. Each such operator acts on subspace π_i as follows. It outputs a k_i-dimensional subspace of π_i randomly if $\dim(\pi_i) > k_i$ and does not change π_i otherwise. For d-dimensional codeword subspaces,define $r_i = (d - k_i)^+$ to be the maximum number of erasures introduced by the channel on π_i. Let also $t \triangleq \dim(\pi_E)$.

THEOREM 2. *Consider a set of orthogonal identifiable codebooks $\mathcal{C}_i$ used over the channel in (9), where $\pi_i \in \mathcal{C}_i$, $1 \le i \le n$, and π_R is the received subspace. If*

$$2(t + \sum r_i) < D(\mathcal{C}), \tag{10}$$

then a minimum distance decoder will correctly decode all transmitted subspaces. The codebook $\mathcal{C}$ is defined as: $\mathcal{C} = \{\pi_1 + \cdots + \pi_n : \pi_1 \in \mathcal{C}_1, \ldots, \pi_n \in \mathcal{C}_n\}$ and $D(\mathcal{C})$ as: $D(\mathcal{C}) = \min_{i:\ 1 \le i \le n} D(C_i)$.

PROOF. Denote by $\pi = \pi_1 + \cdots + \pi_n$ the overall transmitted subspace, and by $\pi' = \pi'_1 + \cdots + \pi'_n$, where $\pi'_i = \mathcal{H}_{k_i}(\pi_i)$. Then

$$d(\pi_R, \pi) \overset{(a)}{\le} d(\pi_R, \pi') + d(\pi', \pi) \overset{(b)}{\le} \sum_{i=1}^{n} r_i + t,$$

where (a) follows from the triangle inequality and (b) follows because the codebooks $\mathcal{C}_i$ are disjoint. For $\pi_W \neq \pi$, $\pi_W \in \mathcal{C}$,

$$D(\mathcal{C}) \le d(\pi, \pi_W) \le d(\pi, \pi_R) + d(\pi_R, \pi_W).$$

Combining these two inequalities we can write

$$d(\pi_R, \pi_W) \ge D(\mathcal{C}) - (t + \sum_{i=1}^{n} r_i).$$

Given (10) a minimum distance decoder chooses π. □

We mention without proof that if $2(t + r_i) < D(\mathcal{C}_i)$, then a minimum distance decoder will correctly decode Source i.