Identity Aware Sensor Networks

Lorenzo Keller, Mahdi Jafari Siavoshani, Christina Fragouli, Katerina Argyraki, and Suhas Diggavi
School of Computer and Communication Sciences, EPFL, Switzerland

***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 occured, 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 should communication happen in such identity-aware sensor networks. We re-examine the traditional source-identity/message separation and propose a scheme for jointly encoding the two. We use this to develop a communication method for identity-aware sensor networks and show it to be energy efficient, simple to implement, and gracefully adaptable to scenarios frequently encountered in sensor networks—for instance, node failures, or large numbers of nodes where only few are active during each reporting round.

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 (*i.e.*, in this context, an identity represents a specific location); 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 measurement. 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 measurement) 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 sensor measurements into one value (*e.g.*, their sum)—as typically done in identity-unaware networks. Second, nodes located close to the sink have to forward much 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 (*i.e.*, more expensive) to charge the network through solar energy or other natural resources.

With this work, we address the question of how should communication happen in identity-aware sensor networks—to the best of our knowledge, this question has not been addressed before, although such networks are useful in practice (see §II-A). We start by observing 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. Based on this observation, we develop a new communication protocol for identity-aware sensor networks that is energy efficient, but also simple to implement and inherently adaptable to scenarios frequently encountered in sensor networks—for instance, node failures, or large numbers of nodes where only few are active during each reporting period.

Our method is distinguished by two key elements:

- Instead of specifying its identity and measurement in separate fields, each sensor *jointly encodes* the two using a set of fixed-size vectors.
- The encoding is such that intermediate nodes can *combine multiple incoming vectors* into a single one (of the same fixed size) and forward only that toward the sink.

The basic idea is to assign a different codebook to each sensor and let each sensor implicitly convey its identity through its choice of codebook, as illustrated in §II-B. 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. We describe our coding scheme at a high level in §II and more formally in §III.

We show that the resulting communication method has the following properties:

- *Energy efficiency and balancing*: We look to minimize the number of frames and bytes transmitted by each sensor. Moreover, each sensor performs approximately the same amount of frame and byte transmissions, independently from its position in the network.
- *Low-complexity network operation*: Intermediate nodes perform the same simple operations (vector combinations) irrespective of their position in the network and thus transparently to topology changes.
- *Error resilience and adaptability*: Our method gracefully incorporates protection against errors and node failures. Moreover, it allows end-to-end adaptation to specific application needs without perturbing intermediate-node operation.

We close with some preliminary experimental results obtained using the TOSSIM [2] simulator (§IV).

Related Work

Recently, techniques inspired from coding and network coding have been successfully used to harness the broadcasting capabilities of the wireless medium [3], [4], [5], [6], [7], implement intelligent in-network storage [8], and provide resilience in lossy environments [9], [10]. 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 and 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 [11], quantum communication [12], and, more recently, network coding, to provide error and erasure correction [13]. All 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.

Significant research effort has been invested in reducing sensor-communication overhead through distributed, in-network data aggregation. The proposed techniques exploit data correlation to perform compression [14], [15], [16] or calculate functions of the observed measurements such as their average [16]. 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, all the received identities and messages into a single packet. As we discuss 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 (see §III-A, Example 3); 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.

II. Sensor Identification

A. Applications

We consider sensor networks where each node needs to communicate (1) its identity and (2) 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 communicated 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 [17].

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 [17].

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 [17], [15]. 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 [14] to reconstruct the entire spatial field.

Multi-stage Collection: Sometimes we are not interested in reconstructing an entire spatial field, 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 [18], [19]. 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.

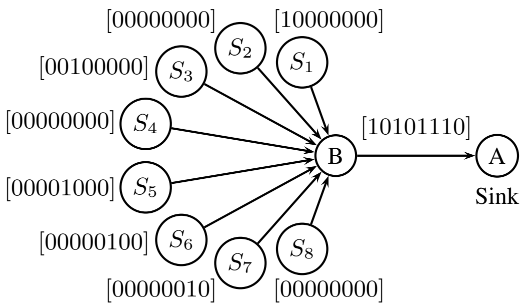


Fig. 1. Sources $S_1, \ldots, S_8$ send their id and a one-bit message to the sink A through a relay node B.

Consider the simplified network of Fig. 1, where 8 source nodes communicate information to a sink node A via a relay node B. Suppose each source $S_i, i = 1, \ldots, 8$, wants to communicate a 1-bit message to the sink A. Each source creates a packet that contains 3 bits specifying its identity and a 1-bit 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 in a single packet. We call this communication protocol *packet aggregation*; it can be easily extended to work on an arbitrary tree: each source sends out a packet with an identity and a message specified in separate fields; intermediate nodes aggregate and forward incoming packets toward the sink. Packet 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: 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 Fig. 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 extended to work on an arbitrary tree: source S_i sends 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. Compared to packet aggregation, this "coding-based" protocol 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 information per packet. Note that 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.

Although simple, the tree of Fig. 1 captures the behavior of the two protocols on all trees (with an arbitrary number of nodes) that connect 8 sources to a sink through a single link; for instance, in the 14-node tree of Fig. 2, packet aggregation would result in 32 bits being transmitted over link BA, whereas the coding-based protocol would result in 8 bits being transmitted over the same link, exactly as in the tree of Fig. 1.

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 Fig. 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 ([20], Chapter 14).

Note that in our joint identity-message coding protocol, 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, thus making it ideally suited for environmentally powered sensor networks.

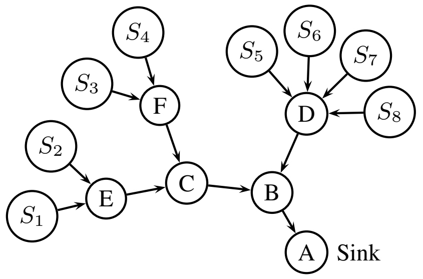


Fig. 2. A tree with 8 sources.

The simple coding-based protocol we have described illustrates the basic idea and benefits of joint identity-message coding, but is specialized to the case where all sources communicate information at about the same time (such that intermediate nodes can combine their packets) and convey a single value to the sink. We present our general code design in §III.

C. Definitions and Assumptions

Subspaces: In our communication protocol, information is transferred through the exchange of vectors of length ℓ, *i.e.*, vectors that belong to the ℓ-dimensional vector space $V = \mathbb{F}_q^\ell$. A *subspace*, denoted by π, is a subset of the vector space V, which is a vector space itself [21]. We say that two subspaces are *distinct* if they differ in at least one dimension. The number of distinct d-dimensional subspaces of $\mathbb{F}_q^\ell$ equals

$$\mathcal{G}_q(\ell,d) \triangleq \left[\begin{array}{c} \ell \\ d \end{array}\right]_q = \frac{(q^\ell-1)\cdots(q^{\ell-d+1}-1)}{(q^d-1)\cdots(q-1)}, \quad (1)$$

and is called the Gaussian number. We use two subspace operations:

- $\pi_\alpha + \pi_\beta = \{x+y \mid x \in \pi_\alpha, y \in \pi_\beta\}$ is the *sum* of subspaces π_α and π_β, *i.e.*, the subspace spanned by the vectors in π_α and π_β;
- $\pi_\alpha \cap \pi_\beta = \{x \mid x \in \pi_\alpha \text{ and } x \in \pi_\beta\}$ is their *intersection*, *i.e.*, the largest subspace that belongs to both π_α and π_β.

Moreover, we need a quantitative metric of "how far apart" two subspaces are; we define the *distance* between two subspaces π_α and π_β as

$$d(\pi_\alpha,\pi_\beta) \triangleq \dim(\pi_\alpha+\pi_\beta) - \dim(\pi_\alpha \cap \pi_\beta). \quad (2)$$

Network Operation: We consider a network of n sensors (hence, n potential sources) and a sink, where the nodes have formed a spanning tree rooted at the sink. Our network operates in rounds; in each round, the following takes place:

- *Encoding*: Source i potentially communicates to the sink its identity and a message from a small set $\mathcal{M}_i$. For this, it uses codebook $\mathcal{C}_i$, which maps each message in $\mathcal{M}_i$ to a d-dimensional subspace[1] of $\mathbb{F}_q^\ell$, *i.e.*,

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

  To communicate a certain message, node i chooses the corresponding subspace π from $\mathcal{C}_i$ and generates d vectors that span π.
- *Relaying*: A node without children sends its vectors to its parent; a node with children linearly combines the vectors generated by itself with the vectors sent by its children into d new vectors, and sends these to its parent.
- *Decoding*: Like any other node, the sink combines the vectors sent by its children into d new vectors. These are linear combinations of the vectors generated by the sources—in particular, if source i generates vectors that span subspace π_i, the sink observes vectors from subspace $\pi_1 + \pi_2 + \cdots + \pi_n$. The sink then uses these d vectors and the codebooks $\{\mathcal{C}_i\}$ to decode the identities of all the sources along with the message sent by each source.

We can represent the set of d vectors, $\{\mathbf{y}_j\}$, produced by the sink, as

$$\mathbf{Y} = \begin{bmatrix} \vdots \\ \mathbf{y}_j \\ \vdots \end{bmatrix} = \sum_{i=1}^{n} \mathbf{G}_i \mathbf{X}_i$$

where $\mathbf{X}_i$ is a matrix that has the original vectors sent by source i as its rows, and $\mathbf{G}_i$ is a "mixing matrix" that represents the end-to-end transformation of $\mathbf{X}_i$. A key point of our approach is that 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 parties. A source can only communicate information to the sink, because it represents each message with a subspace, and subspaces are unaffected by linear operations.

[1]In principle, we can use subspaces of different dimensions, but, for simplicity, in this paper, we restrict our code design to subspaces of equal dimension.

TABLE I
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	$\pi_1+\pi_6$	$\pi_2+\pi_6$	$\pi_3+\pi_6$

Identifiability: We define a *code* as a set of n codebooks, each corresponding to a source. We say that a code is *identifiable*, if every possible combination of subspaces generated by the n sources using this code results in a *distinct* union subspace. More formally:

*Definition 1 (**Identifiable Code**):* An identifiable code is a set of n codebooks $\mathcal{C}_i = \{\pi_j^{(i)} : 1 \le j \le |\mathcal{M}_i|\}$, $i = 1,\ldots,n$, with $\pi_j^{(i)} \subseteq \mathbb{F}_q^\ell$ such that we have $\pi_{i_1} + \cdots + \pi_{i_n} \neq \pi_{j_1} + \cdots + \pi_{j_n}$ when[2] $(i_1,\ldots,i_n) \neq (j_1,\ldots,j_n)$ and π_{i_k} and π_{j_k} are the subspaces chosen by source k.

Example 1: Consider two sources, S_1 with codebook $\mathcal{C}_1 = \{\pi_1,\ \pi_2,\ \pi_3\}$, and S_2 with codebook $\mathcal{C}_2 = \{\pi_4,\ \pi_5,\ \pi_6\}$. Table I summarizes all possible subspace combinations that can be generated when each of the two sources communicates a message to the sink. For this code to be identifiable, we want all 9 entries in Table I to correspond to distinct subspaces. □

III. Joint Identity and Data Coding

We design codes that have the following properties:

1) **Identifiability:** They allow sources to convey their identities and messages to the sink.
2) **Error resilience:** They gracefully incorporate protection against packet loss and corruption.
3) **Adaptability:** They are easily adaptable to specific application needs.

We address each of these properties in turn.

A. Identifiable Codes

We start from the observation that, in many practical scenarios, the network topology as well as the application impose a natural limit on the number of source vectors that get combined within the network. Indeed, two source vectors (generated by two different sources) get combined only if their paths to the sink overlap: In Fig. 2, the sink has a single neighbor (node B), so all the source vectors 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 Fig. 2 as the "branch" of the network corresponding to one

[2]This inequality simply means that there is at least one source k such that $i_k \neq j_k$.

of these neighbors; in this case, each packet that reaches the sink contains approximately $\frac{n}{k}$ combined source vectors. Moreover, in many applications (for instance, anomaly-sensing applications), we expect only a subset of the sensors to report during each round, hence, fewer than n sets of source vectors to get combined within the network. Finally, we can also enforce a strict upper limit on the number of source vectors that get combined—if we append to each vector a few bits that count the number of combined source packets it contains.

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 source vectors from *at most* m sources. We want to design codes that allow the sink to look at each received vector and determine (1) which is the corresponding subset of sources and (2) what are their messages. Note that $m = n$ corresponds to the special case where all sources are active and all source vectors are 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). We later generalize to an arbitrary message-set size.

Single-bit Messages: Our construction proceeds as follows: select a linear code of length n, minimum distance $d_{min} = \min\{2m+1, n+1\}$, and redundancy ℓ, with ℓ as small as possible; consider the $\ell \times n$ parity check matrix $\mathbf{H}$ [22]; 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.

Identifiable Codes for Single-bit Messages:

- Let $\mathbf{H}$ be the $\ell \times n$ parity check matrix of a binary code with minimum distance $d_{min} = \min\{2m+1, n+1\}$
- Source i uses $C_i = \{< h_i >\}$, where h_i is a column of $\mathbf{H}$

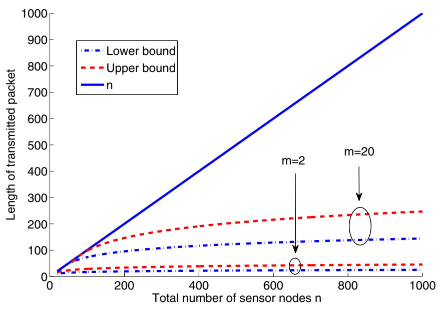


Fig. 3. Bounds on the length ℓ of generated binary vectors ($q = 2$) when $m = 2$ and $m = 20$ sources get combined, as a function of the number of sensor nodes n.

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 set of* $d_{min} - 1$ *columns of the parity check matrix* $\mathbf{H}$ *are linearly independent* [22]. For example, for $d_{min} = 2m+1$, any $2m$ columns of the parity check matrix are linearly independent; thus, if at one round we have $m_1 \leq m$ active sources, each sending a different vector u_i, and at another round we have a different set of $m_2 \leq m$ active sources, each sending a vector v_j, then

$$v_1 + v_2 + \cdots + v_{m_1} \neq u_1 + u_2 + \cdots + u_{m_2} \tag{3}$$

This inequality is a direct consequence of the fact that any $2m$ vectors are linearly independent: indeed, if (3) was an equality, there would be $2m$ 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 (3). This implies that, for all the cases where $n \geq m \geq \frac{n}{2}$, *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, which corresponds to the simple code of §II-B.

The scalability of our code depends on how ℓ (the size of the vector generated by each source) scales with n (the size of the network) and m (the maximum number of vectors that can get combined). This is related to a well studied problem in coding theory, namely, for a given code length n, 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 [22]. Using the Gilbert-Varshamov lower bound and the sphere packing upper bound [22], for $m < \frac{n}{4}$ we get that

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

where $H_q(\cdot)$ is the q-ary entropy function, namely, $H_q(p) = p\log_q(q-1) - p\log_q p - (1-p)\log_q(1-p)$. It is easy to see that for fixed values of m the upper and lower bounds behave as $\mathcal{O}(m \log n)$ as n grows. Fig. 3 plots the bounds from (4) as a function of n, for $m = 2$ and $m = 20$. 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 $\frac{d_{min}-1}{n} = \frac{2m}{n}$ 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.

Example 2: Using a table of the best known codes [22], we can see that there exist binary linear codes of length $n = 512$ with redundancy $\ell = 18$ and minimum distance $2m+1 = 5$. This means that, in a sensor network with $n = 512$ nodes, if at most $m = 2$ source vectors get combined, we need to use vectors of length $\ell = 18$. $\square$

General Case: We now consider the general case, where each source communicates one of $|\mathcal{C}_i|$ 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 $|\mathcal{C}_i| = q^{\Delta} - 1$ 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 [13], using Π_i as our original space.

Identifiable Codes:

- Let $\mathbf{H}$ be the $\ell \times n\Delta$ parity check matrix of a binary code with minimum distance $\min\{2m\Delta + 1, n\Delta + 1\}$
- Assign to source i the subspace Π_i spanned by the $(i-1)\Delta + 1$ to $i\Delta$ columns of $\mathbf{H}$, $i = 1, \dots, n$
- Sensor i uses $\mathcal{C}_i = \{\pi_i \mid \pi_i \subset \Pi_i, \dim \pi_i = 1\}$

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 m vectors can get combined, the sink will receive vectors that are linear combinations of at most $m\Delta$ columns of $\mathbf{H}$. Thus, provided the minimum distance of the code is greater than $2m\Delta + 1$, two received vectors will be equal if and only if the set of active users and their messages 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.

Example 3: We now illustrate the theoretical benefits of joint identity-message coding over packet aggregation, with respect to the maximum amount of energy consumed per sensor. Consider a tree, similar to the example in Fig. 2, where n sources connect to a sink A through a single link BA. Assume that at most m sources are active, each communicating a single-bit message. Packet aggregation requires $m \log_2 n$ identity bits to traverse the link BA, while coding requires a number of bits ℓ bounded according to (4). Fig. 4 shows that, with aggregation, the load on link BA is significantly larger in the case of packet aggregation than with coding.

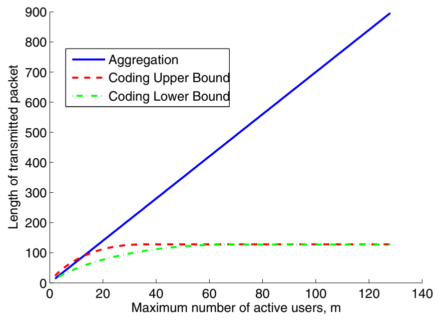


Fig. 4. Comparison of the maximum transmission load for aggregation and coding as a function of m and for $n = 128$.

The difference can be interpreted as follows: With coding, to each set of m sources corresponds a specific coded packet that is received by the sink. With aggregation, the m source packets may be aggregated in an arbitrary order, and all $m!$ possible permutations convey the same message to the sink; because this ordering conveys no information, we lose $\log_2 m!$ bits, which equals the gap between the coding lower bound[3] and aggregation in the plot. □

B. Error Resilience

In real networks, packets get lost due to congestion, corruption, or failures. One method to deal with such scenarios is to rely on MAC-layer retransmissions to provide error resilience. Alternatively, our code construction can be naturally extended to provide forward error correction (which, unlike MAC-layer retransmissions, does not require feedback). Such an approach is well matched to the cases where feedback cannot be readily used or sensors fail (and could not retransmit anyway).

Our general code construction allocated one Δ-dimensional subspace Π_i to each source i. Instead, we now use d-dimensional subspaces within Π_i and, moreover, construct each codebook $\mathcal{C}_i$ to *no longer contain all* $\mathcal{G}_q(\Delta, d)$ distinct d-dimensional subspaces within Π_i, but only a set of d-dimensional subspaces that are "far apart." The idea is to introduce redundancy into the transmitted information, by separating the subspaces chosen as codewords by a certain "distance."[4] We define the minimum distance of the codebook $\mathcal{C}_i$ as the closest two subspaces from this codebook can get. More formally,

$$D(\mathcal{C}_i) \triangleq \min_{\pi_\alpha, \pi_\beta \in \mathcal{C}_i : \pi_\alpha \neq \pi_\beta} d(\pi_\alpha, \pi_\beta), \tag{5}$$

where $d(\pi_\alpha, \pi_\beta)$ was defined in (2).

Identifiable Codes for Error Protection:

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

We provide without proof the following theorem, and refer the interested reader to the extended version of this paper [23] for a detailed discussion of the proof and the general scheme.

Theorem 1: Consider a set of codebooks $\mathcal{C}_i$ used over a channel that erases r_i vectors from source i; moreover, assume that t corrupted vectors are injected in the network. If

$$(2r_i + t) < D(\mathcal{C}_i), \tag{6}$$

where $D(\mathcal{C}_i)$, defined in (5), is the minimum distance of the codebook $\mathcal{C}_i$, then the sink can successfully determine whether source i was active and recover its message. □

[3]The currently best found codes closely follow the lower bound.

[4]Note that traditional erasure correcting codes (like the Reed-Solomon code [22]) would not work in our case, since they would *not* be oblivious to linearly mixing packets generated by the *same* source.

TABLE II
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

C. Adaptability

Different sensor-network applications impose different requirements on our code design; we now illustrate through examples how we can adapt our codes to suit such requirements in an end-to-end fashion, *i.e.*, *without* changing the relaying operation at intermediate nodes. Such flexible and network-transparent operation is not possible with packet aggregation and is a distinct advantage of our architecture. We discuss two specific applications, data-dependent identifiability and sensor clustering, and refer the reader to [23] for additional applications.

Data-dependent Identifiability: Suppose that we are interested in sensor identities only when certain combinations of measurement data occur, *e.g.*, we care to know the sensor identities only when there are discrepancies in their observations.

Example 4: Consider two sources, each observing one of three possible values in the set $\mathcal{M} = \{0, 1, 2\}$. Source S_1 employs codebook $\mathcal{C}_1 = \{\pi_1,\ \pi_2,\ \pi_3\}$, while source S_2, codebook $\mathcal{C}_2 = \{\pi_4,\ \pi_5,\ \pi_6\}$. We are interested in their identities, only when their observations do not match. Hence, we need to implement the function specified in Table II, according to which, when the two sources observe the same value, the sink receives the same subspace, no matter what the observed value is.

To implement this function we use vectors of size $\ell = 2d$, and d-dimensional subspaces of $\mathbb{F}_2^\ell$ as codewords. 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 $\pi + \bar{\pi} = \mathbb{F}_2^\ell$. We 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 observations of sources S_1, S_2. Our code is identifiable, because $\pi_{i,i} = \mathbb{F}_2^\ell$ and, by construction, $\pi_i + \bar{\pi}_j$ is distinct for $i \neq j$. Note that each individual source does not need to know what the other source has observed. $\square$

Sensor Clustering: Consider a densely deployed sensor network that measures a certain spatial field; suppose we want only care to coarsely divide this field into regions of interest. In this case, it makes sense to cluster sensors into groups; by assigning to all the sensors in each group the same codebook, the sink can distinguish whether any sensor in a given group observed a given value, but not the number of nodes observing that value. Once this coarse characterization of the spatial field is complete, we can explore areas of interest at a better granularity by asking the sensors in this area to switch to identifiable codebooks.

Other Adaptations: In this paper, we only considered sensor networks with a single sink, which create a tree topology to connect the sources to the sink. However, networks can have more than one sinks or employ multi-path routing for increased error resilience. Our codes are oblivious to network structure and the number of sinks—indeed, this is one of the strengths of our design.

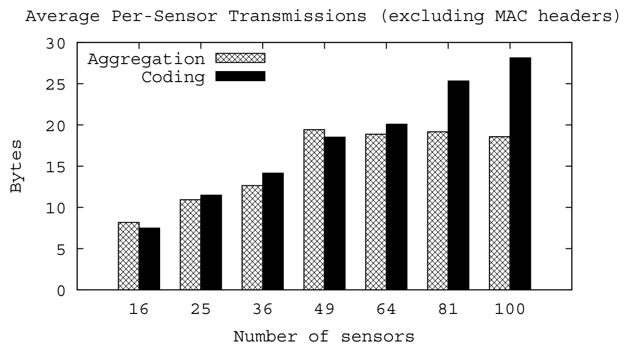
IV. Simulations

Setup: We used our joint identity-message coding scheme to implement a data-collection protocol that operates in rounds: The nodes first use the collection tree protocol (CTP) [24] to build a spanning tree rooted at the sink. In each round, each node produces a message, jointly encodes it with its identity, and sends the resulting vector(s) to its parent; each node waits to collect vectors from all its children, linearly combines (*i.e.*, XORs) them with its own vector(s), and propagates them further upstream. Hence, in each round, each node produces a set of fixed-size vectors, whose size depends on the particular code used. We will refer to this protocol as "coding-based collection."

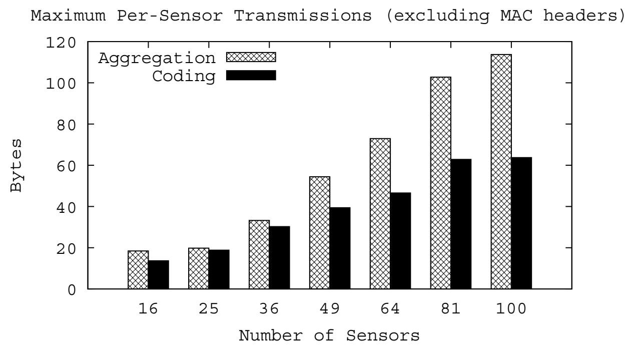
As a baseline for comparison, we implemented a similar tree-based data-collection protocol that uses aggregation: instead of linearly combining the data sent by its children, each node packs that data into a sequence of {identity, message} pairs. Hence, in each round, each node produces a variable-length sequence, whose size depends on the node's position on the tree—nodes that are closer to the sink produce longer sequences.

We implemented both protocols as TinyOs [25] modules and tested them with the TOSSIM simulator [2]. We present some preliminary results regarding the energy-efficiency of the two protocols in the context of a simple application, where sensors produce single-bit messages, all sensors are active in every round, and vectors from all sensors may be combined; we should note that this is the worst-case scenario for our coding scheme, which was designed assuming that vectors from only a subset of the sensors can be combined. For this application, coding-based collection uses the identifiable codes for single-bit messages defined in III-A; in an n-node network, this results in each node transmitting one n-bit vector per round. We are currently working on extending our simulations to cover more applications, as well as demonstrate the error resilience of the two protocols.

Preliminary Results: Fig. 5(a) shows the average per-node, per-round transmissions for the two protocols (*i.e.*, we count the total number of bytes transmitted during the experiment and divide them by the number of rounds and nodes); this is useful in determining the overall energy consumed by the network. Fig. 5(b) shows the average per-round transmissions performed *by the most burdened node* (*i.e.*, we count the average per-round bytes transmitted by each node and report the highest number); this is useful in determining the period of maintenance of the network (how often a battery needs to be changed) or the minimum per-sensor energy required in networks periodically recharged through natural resources. Figs. 6(a) and 6(b) show a detailed breakdown for the same numbers. We show our results in terms of bytes, not frames,

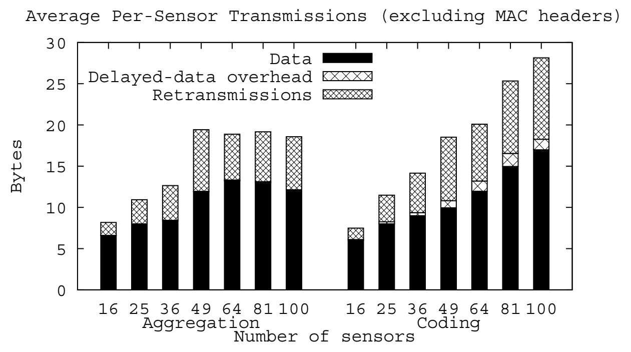


(a) Total number of transmitted bytes divided by the number of rounds and nodes

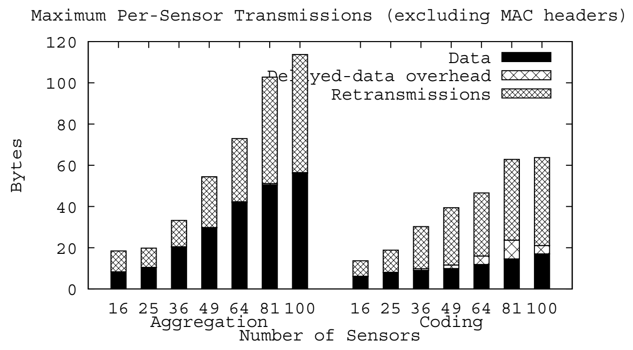


(b) Maximum number of average per-round bytes transmitted by any node

Fig. 5. Average and maximum per-node transmissions excluding MAC headers. The nodes are placed on a square grid, 8 meters from one another. We chose this density, because it allows the CTP protocol to operate without end-to-end loss. We stop at $n = 100$ nodes, because connecting more nodes to a single sink introduced contention and end-to-end loss independently from the chosen density. Simulator parameters are set according to an outdoors environment. For each n we ran our protocols for 1000 rounds.



(a) Breakdown of the total number of transmitted bytes divided by the number of rounds and nodes



(b) Breakdown of the maximum number of average per-round bytes transmitted by any node

Fig. 6. Fig. 5 breakdown. "Data" corresponds to packets that were properly aggregated or combined. "Delayed data" corresponds to packets that were not aggregated or combined with other packets, because they reached the corresponding node *after* it sent out its sequence/vector; such delays are a result of topology changes. "Retransmissions" corresponds to packets retransmitted due to link-layer errors.

as the two protocols lead to similar frame transmissions; also, in Figs. 5 and 6, we do not take into account MAC-header and acknowledgment overhead, as they are the same for both protocols—we discuss these overheads later.

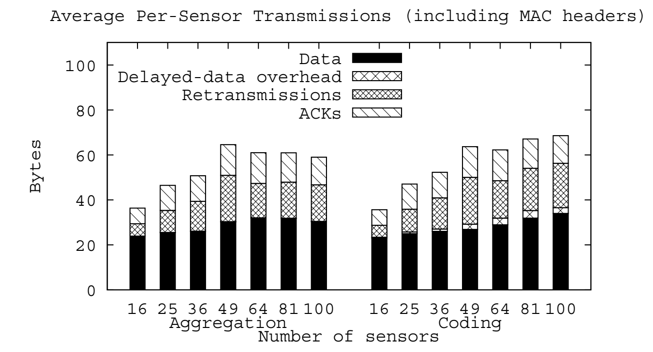
As expected, aggregation-based collection performs better in terms of the total number of bytes transmitted by the network. On the other hand, coding-based collection performs significantly better in terms of the maximum number of bytes transmitted by any single node. For instance, according to Figs. 6(a) and 6(b), in a 64-node network that is recharged through natural resources, aggregation-based collection requires more sophisticated recharging equipment than coding-based collection: the former imposes 65% more byte transmissions than the latter on the most burdened node—ignoring, for the moment, the MAC-layer overhead.

A closer look (Fig. 6(b)) reveals that, for coding-based collection, maximum per-node transmissions are dominated by link-layer retransmissions, *i.e.*, the more burdened nodes are the ones that have to retransmit more frequently due to weak connectivity—which are often located on the periphery of the network. In contrast, for aggregation-based collection, the most burdened nodes are the ones that need to convey the most data, *i.e.*, the nodes located closest to the sink; these nodes are most important for the operation of the network, yet they would be the first to fail from battery depletion.

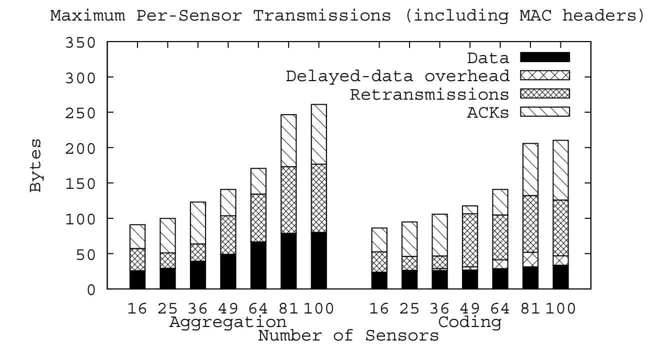
It is also worth noting that, in the case of coding-based collection, the values for the average and maximum per-node transmissions would be exactly the same, if it weren't for retransmissions and topology changes that introduce delayed transmissions (Fig. 6). Still, they are significantly closer than in the case of aggregation-based collection, suggesting uniform energy consumption across the network.

Link-layer Overheads: Fig. 6(b) shows that coding significantly outperforms aggregation in terms of maximum per-node *data* transmissions. However, with our current implementations, the gap is reduced by retransmissions and MAC headers—17 bytes for data frames and 11 bytes for acknowledgments. Figs. 7(a) and 7(b) show the performance of the two protocols when taking into account these headers.

We should note that these overheads are not fundamental to our joint identity-message coding and can be removed: Acknowledgments and retransmissions are due to our CTP-based implementation, which relies on the link layer for reliable communication; we plan to replace them with multi-



(a) Breakdown of the total number of transmitted bytes divided by the number of rounds and nodes



(b) Breakdown of the maximum number of average per-round bytes transmitted by any node

Fig. 7. Average and maximum per-node transmissions including MAC headers. These numbers correspond to the same experiments depicted in Figs. 5 and 6, but take into account the 17-byte MAC-layer header for data packets and 11-byte acknowledgments.

path communication—*i.e.*, send each piece of information through multiple paths, such that, even if a path fails, the message reaches the sink with a high probability. Unlike aggregation[5], coding-based collection can use multi-path communication and still be energy efficient, as each node always transmits the same number of fixed-size vectors independently from the number of paths. Moreover, for networks where the measurement reported by each sensor consists of a few bytes, the 17-byte header dictated by the IEEE 802.15.4 frame format becomes inappropriate as it dominates transmission cost; for such applications, it makes sense to develop a lighter link-layer protocol. Addressing these issues is part of our future work.

V. Conclusions

We have formulated the paradigm of identity-aware sensor networks to capture applications where, as illustrated in §II-A, 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 (1) equally balancing the transmission load across all nodes in the network, which is important for networks that are periodically recharged through natural resources; (2) low complexity network operation; (3) graceful incorporation of error resilience and flexible adaptation to specific application needs that is transparent to the operation of intermediate nodes.

[5]Using aggregation with multi-path communication results in multiple copies of the same data in different parts of the network, significantly increasing traffic load.

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] 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.
[4] S. Katti, S. Gollakota, and D. Katabi. Embracing Wireless Interference: Analog Network Coding. In *ACM SIGCOMM*, 2007.
[5] 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.
[6] S. Zhang, S. Liew, and P. Lam. Physical Layer Network Coding. In *ACM MOBICOM*, 2006.
[7] 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.
[8] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth Codes: Maximizing Sensor Network Data Persistence. In *ACM SIGCOMM*, 2006.
[9] Z. Guo, P. Xie, J.-H. Cui, and B. Wang. On Applying Network Coding to Underwater Sensor Networks. In *ACM International Workshop on Underwater Networks*, 2006.
[10] M. Ghaderi, D. Towsley, and J. Kurose. Network Coding Performance for Reliable Multicast. In *Military Communication Conference (MILCOM)*, 2007.
[11] 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.
[12] 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.
[13] R. Koetter and F. Kschischang. Coding for Errors and Erasures in Network Coding. In *IEEE International Symposium on Information Theory (ISIT)*, 2007.
[14] S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed Compression in a Dense Sensor Network. *IEEE Signal Processing Magazine*, 19(2):51–60, March 2002.
[15] 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.
[16] 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.
[17] G. Pottie and W. Kaiser. *Principles of Embedded Networked Systems*. Cambridge University Press, 2005.
[18] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model Driven Data Acquisition in Sensor Networks. In *Conference on Very Large Data Bases (VLDB)*, 2004.
[19] D. Tulone and S. Madden. PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks. In *European Workshop on Wireless Sensor Networks*, 2006.
[20] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.
[21] Horn and Johnson. *Matrix Analysis*. Cambridge University Press, 1990.
[22] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library, 1977.
[23] L. Keller, M. Jafari, K. Argyraki, C. Fragouli, and S. Diggavi. Identity Aware Sensor Networks. Technical report, Ecole Polytechnique Federale de Lausanne, 2008.
[24] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo. The Collection Tree Protocol (CTP). http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html.
[25] Tinyos. http://www.tinyos.net/.