Distributed Spatiotemporal Suppression for Environmental Data Collection in Real-World Sensor Networks

William C. Evans, Alexander Bahr and Alcherio Martinoli
Distributed Intelligent Systems and Algorithms Laboratory
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
Email: firstname.lastname@epfl.ch

***Abstract*—Environmental processes are often severely oversampled. As sensor networks become more ubiquitous for this purpose, increasing network longevity becomes ever more important. Radio transceivers in particular are a great source of energy consumption, and many networking algorithms have been proposed that seek to minimize their use. Traditionally, such approaches are often data agnostic, i.e., their performance is not dependent on the properties of the data they transport. In this paper we explore algorithms that exploit environmental relationships in order to reduce the amount of transmitted data while maintaining expected levels of accuracy. We employ a realistic testing environment for evaluating the power savings brought by such algorithms, based on Sensorscope, a commercial sensor network product for environmental monitoring. We implement and test a suppression-based data collection algorithm from literature that to our knowledge has never been implemented on a real system, and propose modifications that make it more suitable for real-world conditions. Using a custom extension board developed for in situ power monitoring, we show that while the algorithms greatly reduce the amount of energy spent on transmitting packets, they have no effect on the real system's overall power consumption due to its preexisting network architecture.**

I. Introduction

Environmental processes are often severely oversampled. As sensor networks become more ubiquitous for this purpose, increasing network longevity becomes ever more important. Radio transceivers in particular are a great source of energy consumption, and many networking algorithms have been proposed that seek to minimize their use. Traditionally, such approaches are often *data agnostic*, i.e., their performance is not dependent on the properties of the data they transport. In our work, we strive to push the envelope by exploring algorithms that exploit environmental relationships in order to reduce the amount of transmitted data while maintaining expected levels of accuracy. By intelligently suppressing redundant sensor data, we inadvertently characterize the environmental process under measure. As the performance of such techniques improves, we hope to also contribute new information on the behavior of environmental phenomena.

In this paper, we describe the development of a testing environment for evaluating the power savings brought by various approaches to efficient monitoring. Our system is based on Sensorscope, a commercial sensor network product for environmental monitoring, and includes realistic simulation in conjunction with an outdoor testbed. We implement and test a suppression-based data collection algorithm from literature, known as Constraint Chaining [1], that to our knowledge has never been implemented on a real system. We propose extensions to this algorithm that make it more suitable for deployment under particularly dynamic conditions, and detail its behavior in calibrated simulation. Our outdoor testbed integrates a custom developed power monitoring board, allowing us to quantify changes in energy consumption as we tweak the network architecture.

II. Related Work

Efficient monitoring techniques can be largely classified as operating on the basis of either spatial or temporal suppression. Both methods attempt to avoid sending redundant data to the sink, preferring instead to have the sink infer these values based on other received data. Temporally focused techniques operate local to a node, examining its history of measured values. The most basic example is simply not reporting a value if it has not changed since the last measure. In contrast, purely spatial approaches seek to suppress a nodes value if it can be inferred given the value of nearby nodes.

The Probabilistic Adaptable Query (PAQ) system is an approach to temporal suppression based on time series forecasting [2]. It uses autoregressive models maintained locally per sensor in order to keep from sending data directly to the sink. Instead, nodes communicate model parameters as necessary in order to keep the sink's predictions within some defined error bound. Tulone and Madden extend this work with their Similarity-based Adaptive Framework (SAF) [3], adding robustness to quick changes in data trends as well as a location-independent clustering technique that allows the detection of redundant nodes.

Many spatial suppression algorithms attempt to detect and deactivate sets of redundant nodes. Prorok et al. study hierarchical network topologies based on spatial clustering [4]. In this approach, cluster heads may choose to prune their children if the part of the monitored field they represent is highly isotropic as defined by some statistically computed threshold. Arici and Altunbasak propose using a first-order

model to determine the predictability of particular nodes [5]. They define some of the nodes in the network as *macronodes* which attempt to fit a plane over their neighbors' positions and data, commanding easily predictable nodes to stop reporting measurements for some period of time.

A third and wholly separate approach, compressed sensing, draws on recent advances in signal processing that have interesting implications for sensor networks. This technique allows the accurate reconstruction of a signal while sampling at a frequency that does not satisfy its Nyquist rate. Note that unlike the above approaches, compressed sensing reduces the number of measurements taken altogether, an obvious advantage when using active sensors, i.e., sensors that require considerable power just to sample. Recent work has shown that compressive sensing is usable on signals typical to environmental sensor networks [6].

Outside of continuous monitoring, many approaches to efficient data collection seek to reduce overall message volume by eliminating uninteresting data in-network. TinyDB [7] provides such functionality, returning sensor data to the sink in response to simple aggregation queries such as SUM or MAX. Other techniques use in-network triggers to decide when data should be sent to the sink. Yang et al. present a Two-Phase Self-Join scheme that accepts complex monitoring queries from the user and informs the sink should an appropriate event be detected [8].

Constraint Chaining (CONCH) is another algorithm that provides real-time sensor data from a network while implementing spatiotemporal suppression [1]. CONCH operates by detecting highly correlated neighbors and monitoring a suitably defined edge constraint between them instead of their individual values. Unlike similar algorithms which may require functionality not provided by the current Sensorscope network architecture (e.g., intra-network or geographic routing), CONCH imposes very few constraints on the system in which it operates. Our approach to efficient monitoring builds on CONCH, as its few system requirements make it applicable to a wider range of real-world systems.

To our knowledge, none of the algorithms listed in this section have undergone implementation outside of a controlled laboratory environment. A key contribution of this work is the implementation and evaluation of a suppression-based monitoring algorithm on an outdoor commercial sensor network widely used for scientific data collection.

III. System Description

This section describes the chosen sensing platform, as well as simulation and real-world testing environments we used to evaluate the algorithms described later in this paper.

A. Sensorscope System

Sensorscope is a commercial platform for environmental monitoring which has seen extensive use in scientific deployments (e.g., [9] [10]). Individual stations are typically deployed with a number of environmental sensors attached in a daisy chained fashion from a central collection and logging board, henceforth referred to as the *datalogger*. The datalogger also features a short-range radio, which it may use to form a multi-hop network with its neighbors, and optionally a long range GSM transceiver for sending data out of the network. Sensor detection and network configuration is performed autonomously when a station is activated, providing out-of-the-box operation, a key feature for environmental scientists. The following section describes Sensorscope's hardware and network architecture as it relates to this project. A broader look at the Sensorscope system can be found in [11], [12].

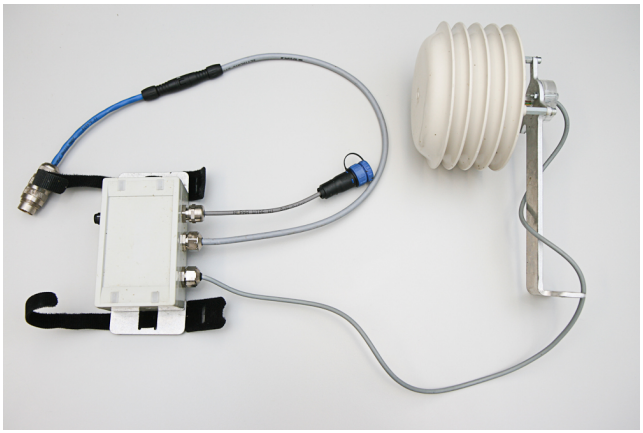

Fig. 1. (left) Sensor module and (right) its attached Sensirion SHT75 digital temperature/humidity sensor.

1) Hardware Description: The stations are built around the datalogger, a central processing board that is responsible for collecting data from its sensor bus while simultaneously maintaining a multi-hop network with its neighbors. The datalogger contains ShockFish TinyNode 584 [13], running a software suite based on TinyOS 2.x. Local communication is performed in the 868 MHz band using a Semtech XE1205 radio transceiver, with a range of up one kilometer given strong line of sight. Optional long range communication is performed via a Telit GM862-GPS GPRS modem. Sensor measurements may be stored to an onboard microSD card. The electronics are protected by an environmentally sealed container with external connectors for the antenna, sensor bus and solar panel.

Each sensor is attached to its own module, separate from the datalogger, which contains its own microprocessor that supports components for sensor sampling and hardware for driving the SPI bus (see Figure 1). This architecture imposes very little constraints on the sensor itself, allowing sampling and intermediate processing to be done asynchronously with the operation of the datalogger. Periodically, all sensors are polled for their latest measurements, which are sent to the datalogger for storage and transmission out of the network.

The stations used in the experiments performed in this paper are deployed with only a single digital temperature/humidity sensor, based on either the Sensirion SHT11 or SHT75 (for availability reasons). The sensor itself is housed in a layered radiation shield that serves to protect the sensor from the elements while still allowing accurate sensing (see Figure 1).

The corresponding sensor module runs on TI's MSP430F149 microprocessor.

2) In-network Algorithm: Sensorscope employs two types of stations: slaves, which communicate only over their short range radio, and masters (i.e., sinks), which collect data from all connected slaves and transmit that data to an off-site server via a cellular network. In the following section we describe the in-network algorithm that enables slaves to forward their data to the sink, and the operation of the long range radio in transmitting data out of the network.

Each station, slave or master, executes the same algorithm for sensor sampling and local communication. The temperature sensor is sampled every 60 seconds. In order to mitigate per-transmission overhead, these sensor values are aggregated into a single packet until either the packet is full (maximum 48 byte payload) or until the packet is ten minutes old (i.e., due to infrequent sampling). Sensor values are timestamped by the datalogger, storing the packet's first measurement's full timestamp (i.e., seconds since the epoch), while only storing the delta with subsequent samples. In our system, in order to get a clear picture of the effect of data suppression, only the temperature is transmitted over the network. Other sensed data, such as network statistics, power consumption, and battery status are exclusively written to the onboard microSD card.

The short-range radio is duty-cycled, turning on every two minutes for a variable window of time. This window is between 4 and 60 seconds long (with two seconds of padding), depending on how much the station and its neighbors usually send each cycle. The window length is adjusted by maximally one second per cycle, until between 60–80% of the frame is spent either sending or receiving packets. Sensorscope uses the Carrier Sense Multiple Access (CSMA) MAC protocol for packet transmission, and all sensor data is sent from station to station in a reliable fashion (i.e., with packet acknowledgements).

Each node with a route to the sink sends a beacon at the beginning of every radio cycle. The beacon contains information for routing, time synchronization and adjustment of the frame length, and is additionally used to estimate link quality with neighboring stations. Each station tracks the number of beacons received by its neighbors during the last 16 radio cycles. A station must have received at least 11/16 of the most recent beacons from a neighbor to consider it for routing purposes.

3) GPRS: Each network may contain multiple master stations, each with a GPRS module for off-site data transmission. Every fifteen minutes, master stations activate their GPRS modules and connect to a preprogrammed server via the Internet. On successful connection, the station authenticates itself to the server using a pre-shared key. It receives a global time reference, and updates its clock accordingly. Data packets are then combined and sent as 3500 byte blocks, but are not interleaved, i.e., each subpacket contains sensor data from a single station maximally spanning over ten minutes, and each subpacket is packed back-to-back. The station waits as long as 10 seconds to receive an acknowledgement from the server before sending its next block.

GPRS operation is done out of phase with the short-range radio. Operating in our suburban environment, the GPRS module typically takes 25–40 seconds after being powered on to connect to the external server, which leaves about between 20–90 seconds for data transmission before the next short-range radio phase begins. We observe a typical speed of 2–3 kbps, allowing about 6000 sensor measurements to be transmitted per cycle per master (assuming a 4 byte payload).

B. Simulation

Simulation is critical to algorithmic performance evaluation for multiple reasons. First, debugging sensor network algorithms is extremely difficult *in situ*. As our testbed is completely wireless and is located outdoors, features such as wireless reprogramming and live debugging are difficult to realize. Second, in order to truly compare different network architectures, it is important that we have an environment that allows us to test algorithmic variations under repeatable (but realistic) circumstances. For the sake of this project, we added simulation support to the Sensorscope codebase using TOSSIM, a realistic simulator for TinyOS [14]. This approach allows us to run the exact same high-level code both in simulation and on the real stations. The simulations in this paper are performed by "replaying" network connectivity and sensor data as logged by the real stations during our outdoor deployments (see Section V-B).

1) Calibration: In order to estimate the efficiency of different network algorithms, we implemented a simple method for simulating power consumed by the radio. We characterized the TinyNode's short-range radio (see Section III-D for details), and input to the simulation the energy required to transmit a packet depending on the length of its payload. We added a layer to the TinyOS radio module that records each duration the radio is turned on along with the size of each packet transmission and the corresponding station's identifier. This calibration process enables us to accurately estimate the per-station radio-related power costs.

2) Server Integration: Under Sensorscope's current network architecture, the only data transmitted from the off-site server to GPRS master stations is a global time reference. While this can easily be faked in simulation, as monitoring algorithms become more complex some out-of-network planning may become necessary (and indeed this is the case for CONCH). For this reason, we implemented a simulated GPRS driver that is able to connect to a normal Sensorscope server via a network socket. This approach allows us to simulate the operation of our modified server simultaneously with the station code.

C. Outdoor Testbed

1) Power Monitoring: While software performance logging allows us to monitor an algorithm's performance (in terms of the reduction in transmitted data, for example), we are ultimately interested in the effect of these algorithms on the network's power budget. In order to directly quantify the effect

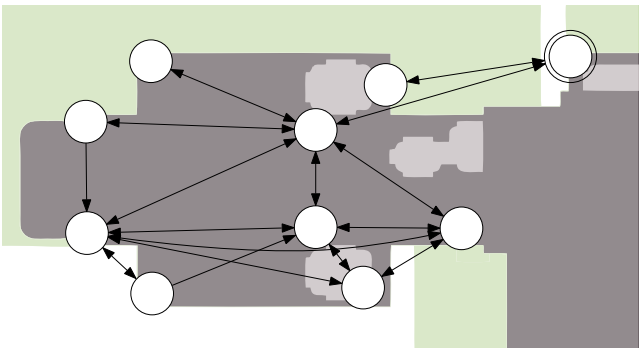

Fig. 3. Map of deployment with high quality links shown. The sink is marked with two concentric circles. Note that some nodes appear very close to each other but do not have a connection– this is often due to vertical separation.

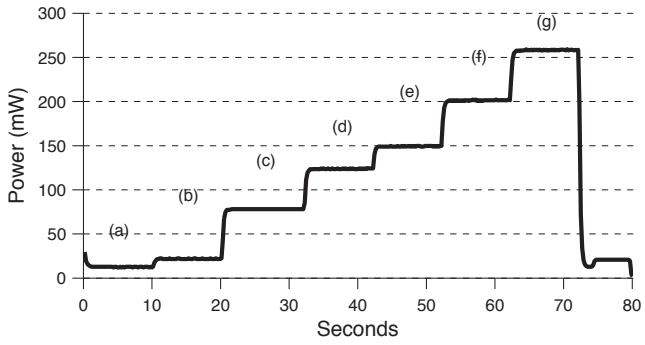

Fig. 4. Power consumed by the datalogger while (a) sleeping, (b) processing, (c) radio on, and transmitting at (d) 0 dBm, (e) 5 dBm, (f) 10 dBm and (g) 15 dBm.

of our algorithms, we have developed an extension board for the Sensorscope datalogger that allows for *in situ* power monitoring, henceforth referred to as the *power board* [15]. The experiments described in this document mark its first successful long-term results.

The power board integrates seamlessly with existing stations. It fits inside the datalogger's enclosure, resting over the the main board, intercepting the ribbon cables connecting its power source and sensor chain (see Figure 2). This approach allows the power board to be used without requiring any mechanical or electrical modification to existing stations. By measuring the current passing through these ribbon cables, the power board is able to monitor the station's complete energy budget, including incoming energy from the stations' solar panel, and energy used by the datalogger, sensor chain, and the power board itself. In addition to power monitoring, the power board is able to disconnect the attached solar panel, allowing the implementation of a software-based charge controller.

The power board has its own microcontroller (TI's MSP430F1611), which allows it to function similarly to other sensors on the bus. It reports measured power data to the datalogger via the SPI bus, as per any other sensor. However, although current measurements are taken at 5Hz, this bus was not intended for streaming sensor data. Thus, the power board reports the mean, minimum and maximum values over each reporting period (one minute in our case). For *ex situ* experimentation, a UART port is included on the power board, allowing one to record higher resolution data as is shown later in this document.

2) Deployment: We deployed nine slave stations and one GPRS master outdoors, around the rooftop of EPFL's GR building and surrounding balconies below (see Figure 3). Each station was equipped with a digital temperature/humidity sensor and power monitoring board. The stations were purposefully placed in such a fashion that the sun passing overhead would introduce changing relationships between neighboring sensors through the day. The short-range radio's transmission power was set to 15 dBm in order to have a power consumption profile similar to a normal deployment. We used 20 dB attenuators attached to each station's antenna in order to limit the range such that a multi-hop network was formed, with a typical maximum distance to the sink of three.

D. Power Characterization

In order to facilitate realistic simulation and to give context to our testbed results, we first characterize the power consumption of the Sensorscope system.

An overview of the cost of various station activities, as

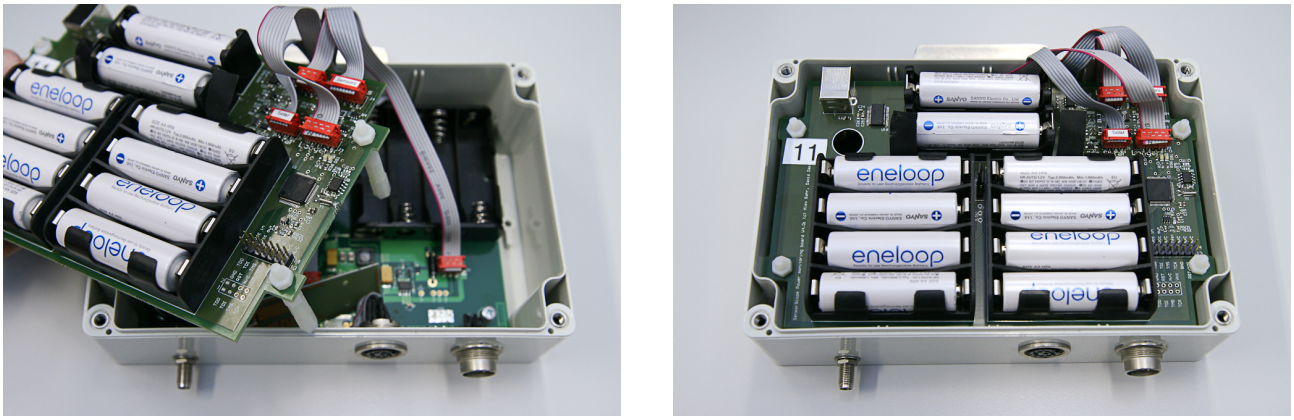

Fig. 2. The power board fits inside the datalogger over the original board.

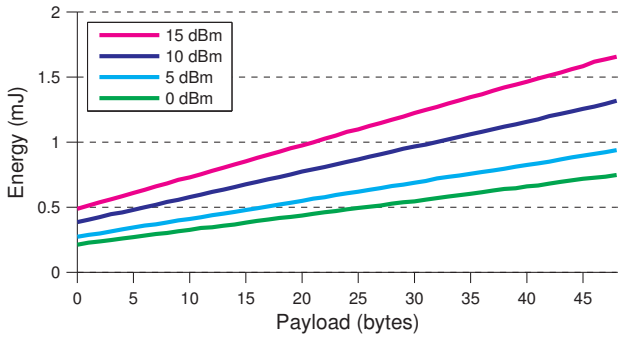

Fig. 5. Energy consumed by TinyNode radio transmission by packet payload size and transmission power. Packets of increasing payload size from 0 to 48 bytes were sent via the TinyNode's Semtech XE105 radio.

measured by the power board, can be seen in Figure 4).

We used a LeCroy WaveSurfer 24Xs-A oscilloscope to measure energy usage during packet transmission, as the power board's maximum effective sampling rate of 5 Hz is far too low to capture it. Due to the presence of large capacitors onboard the data logger, we measured the energy used by the TinyNode in isolation. Using this data we were able to compute the energy used during packet transmission as a function of payload size and transmission power (see Figure 5).

IV. Efficient Data Collection

Exploiting the spatial and temporal relationships inherent to environmental processes is an obvious place to look for energy savings in environmental sensor networks. Simulations over datasets spanning a wide variety of environmental conditions have shown that simply suppressing sensor measurements that have not significantly changed since their last transmission (i.e., by an amount greater than their noise) may reduce the amount of reported data in scientific deployments by more than half [16]. Taking advantage of spatial relationships is much less obvious. In this section, we describe one such spatiotemporal algorithm, and address various difficulties in implementing it as part of an established network architecture. We then propose a modified algorithm which replaces centralized operation in favor of a distributed approach, allowing in-network optimization and increasing robustness in realistic (i.e., unstable) environments.

A. Constraint Chaining

1) Overview: CONCH operates by monitoring the difference between correlated neighbors in the network. The network goes through an iterative training process, during which sensor relationships between neighboring stations are tracked. This data is used to periodically construct a spanning tree over the network such that each edge represents a pair of neighboring stations with values that are highly correlated. This spanning tree is called a *Conch plan*, and its construction is detailed later in this section. Stations that are neighbors in the plan share their sensor values, and transmit only the difference between their values to the sink. These values are never transmitted unless they have changed more than by a user-defined threshold (in our case, we use the SHT75's sensor noise: 0.3 °C). The server is able to reconstruct the values at every station by walking along the aforementioned spanning tree. The flow of data in and out of the network is described in Figure 6.

This approach is well-suited to implementation on Sensorscope stations as it does not require advanced network functionality. Time synchronization, while not strictly necessary, eases implementation by providing a basis by which neighbors may compare their measurements. Some basic neighborhood discovery must be employed in order for stations to coordinate with related nearby nodes. However, most critically, the network must have some centralized means of generating the CONCH plan, i.e., access to the data and neighborhood history of all stations, and a way to then distribute the plan throughout the network. Sensorscope provides all of these features, with the exception of a means by which to send the plan from an off-site server back into the network. We have added naive reliable broadcast in-network and an extra phase during the GPRS cycle for this purpose.

2) Spatial Coordination: Spatial suppression in CONCH is enabled by coordination between pairs of nodes referred to as *updaters* and *reporters*. Each updater/reporter pair monitors one edge in the CONCH plan. Each radio cycle, updaters may transmit their new sensor measurements to a neighboring reporter. The reporter compares its updater's measurement with its own, and if the *difference* between these two values has changed since the last pair of measurements, that difference is transmitted to the sink via normal routing methods.

Packet aggregation works normally as compared to the original Sensorscope algorithm. Both updater and reporter messages are aggregated until a packet is either filled or becomes older than ten minutes. As such, measurements are not assumed to have been suppressed (i.e., because they have not changed) until this time period has passed.

In the case that a node is not found to be correlated with any of its network neighbors, it is given the special role of *direct reporter*. Direct reporters send their absolute sensor values directly to sink (again only when they change by more than a certain threshold).

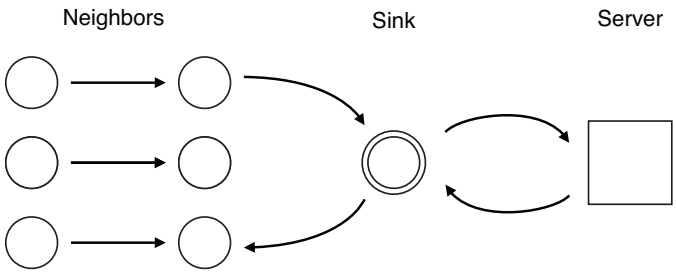

Fig. 6. Data flow in CONCH. Neighboring nodes transmit their sensor difference and neighborhood information to the sink via short-range radio, and the sink sends this data to an off-site server via a cellular link. Periodically, a new CONCH plan is calculated and sent from the server to the sink, which then uses a reliable flooding technique to spread it throughout the network.

3) Plan Construction: We refer to the choice of monitored edges and assignment of updater and reporter roles on each link as the CONCH plan. This structure is calculated periodically at the off-site server, and is based on historical data reported by the sensor network.

The first step is to choose the cheapest set of edges in the network to monitor such that these edges form a spanning tree on the network graph. As the server will only receive the difference in value along each of these edges, if the graph is not connected, the value of disconnected nodes will be unknown. We proceed by assigning a cost to each network edge e equal to $dist(e) * freq(e)$, where $dist(e)$ is the number of hops between the closest node on that edge to the sink, and $freq(e)$ is the number of times the difference along that edge has changed during the previous planning period. Note that as messages are only sent when the difference along an edge changes, this cost represents the number of messages that would be generated in the network if a particular edge were monitored. Direct reporters are chosen by adding a fake edge from each node v directly to the sink, with cost $dist(v) * freq(v)$. To obtain the set of monitored edges, we simply calculate the MST over this graph.

Next we must assign updaters and reporters on each of the edges that do not indicate a direct reporter. Consider that only reporters are responsible for communicating data to the sink, and an optimal CONCH plan will be likely to place reporters closer to the sink than their corresponding updaters. Bearing this in mind, we simply iterate through all monitored edges, marking the adjacent node closest to the sink as its reporter, and the other as its updater. This simplication may be cause excessive energy usage if the node chosen as updater changes value significantly more often than its reporter, however we do not observe such scenarios in our experiments.

We examine the performance of a scheme that executes its replanning phase using the previous N hours of sensor data, repeating this process after another N hours have passed. One could imagine more dynamic schemes in which CONCH plan generation is triggered by some condition detected at the sink, however disseminating a new plan is fairly expensive, so for the sake of simplicity we leave such approaches to future work. In our experiments, we choose $N = 5$ hours.

B. Model-based Constraints

In order to improve the performance of CONCH's spatial and temporal constraints, we replace the basic differential approach described above with a model-based approach. Instead of transmitting sensor readings whenever they have changed over some threshold, we the modify the algorithm to maintain a statistical model of local sensor behavior and instead transmit the model's parameters. Thus, we must only ensure that the sink has accurate model parameters for every node in the network.

More specifically, we employ autoregressive (AR) models, maintained both at each network node and at the sink. AR models are a special case of autoregressive integrated moving average (ARIMA) models. While the general form is more powerful, fitting an ARIMA model is potentially intractable on limited hardware, and may require solving an infinite system of equations. AR models only require solving a strictly linear system of equations, and for small orders can be solved at a low computational cost even on modest hardware (i.e., using Levinson recursion [17]). We choose an AR model of order three (i.e., future values depend on a linear combination of the previous three values) in our work, as it has previously been shown to perform well on environmental datasets while still being computationally feasible in this application [2]. This yields four model parameters: the mean of the last three measured values, and three coefficients corresponding to each value in the window. The exact formulation can be found in [18].

Changes to the underlying algorithm are minimal. Each time a node takes a new measurement, it recalculates its AR model parameters and, if they have changed significantly, transmits the new coefficients along any links on which it is an updater. For each link on which it is a reporter, it then calculates the difference between its own vector of model parameters and those that it received from each updater, and if the difference between any two of these two sets of parameters has changed, they are transmitted to the sink. Note that while we are now operating on four values instead of the sensor's direct value, we have found that the model has excellent predictive capabilities on our datasets, and it nonetheless results in a much higher suppression rate than the original approach when combined with a high planning frequency (see Figure 11).

C. Distributed Constraint Chaining

In its original form, the ability of CONCH to adapt to changes in the network or its environmental conditions is limited by the expense of transmitting a new CONCH plan to every node in the network. Naive reliable flooding requires the transmission of $O(|E|)$ messages, where E is the set of edges on the network graph, and less expensive approaches require additional routing structure to be maintained. Thus we propose Distributed CONCH (DCONCH), in which MST calculation is done in a distributed fashion, allowing individual nodes to quickly adapt to dynamic conditions by making local adjustments to the CONCH plan. In this section, we describe existing approaches to distributed MST calculation, identify a method suitable for our network architecture, and describe CONCH plan generation in DCONCH.

1) Distributed Minimum Spanning Tree: The greatest hurdle in generating the CONCH plan in-network is calculating the MST in a distributed fashion. A number of algorithms have been proposed both for calculating the exact MST (e.g., [19], [20]) and its approximation (e.g., [21]).

Li et al. propose the Local Minimum Spanning Tree (LMST) algorithm for topology control in WSNs [22]. Their algorithm approximates the exact MST as follows: each node in the network calculates the MST over the subgraph formed by its one-hop neighbors. An edge between two nodes exists in the LMST if and only if both nodes have each other in their one-hop MST. This approach has seen considerable acceptance

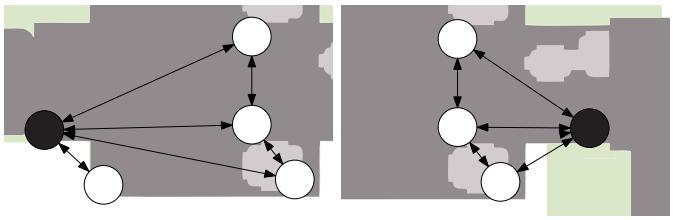

Fig. 7. Examples of one-hop neighborhoods from DCONCH. Reference nodes in black.

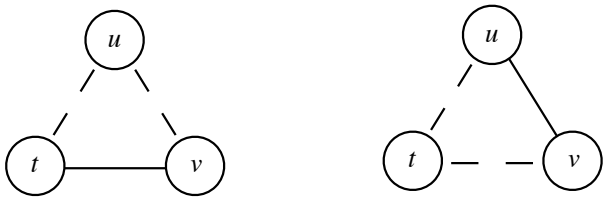

Fig. 8. The contentious edge problem in DCONCH. Consider the network from the perspective of node u (left) and node t (right). Dashed lines represent network connectivity, and solid lines represent plan monitored edges. If the costs on various edges are estimated to be similar, it is possible that nodes in this graph expect the other to take an edge connecting to v. If neither does, v will be orphaned and its value will not be calculable.

in the sensor networks community as it only requires one-hop communication. Cartigny et al. develop an LMST-based approach to reliable flooding, and prove that the LMST of a graph is a subset of the RNG, i.e., it will always more closely approximate the exact MST [23]. Ovalle-Martínez et al. propose a method to convert LMSTs into MSTs by breaking cycles in the network graph [24].

We choose LMST for use in DCONCH as it is well-studied, inexpensive, and frequently used in the context of sensor networks.

2) Implementation: CONCH plan adjustment in DCONCH proceeds as follows. During each radio frame, nodes overhear sensor data sent between their neighbors. These measurements are compared with the eavesdropping station's own data, and it counts the number of times the difference between these values changes. The cost of monitoring the link between these two stations is equal to this value multiplied by the minimum distance between either of these nodes to the sink (note that this information in particular is already exchanged via periodic beacons).

Note that the form in which a station transmits sensor data depends on its role. If the node is an updater or a direct reporter, it will transmit its actual sensor value. However, if a node is a reporter, it will only send the difference between its own value and that of its paired updater. If only the reporter is in range, i.e., the eavesdropping station is unable to overhear the updater's value, it is impossible to determine the reporter's actual value. For this reason, we only consider updater transmissions during edge cost calculation.

One issue in implementing DCONCH is that due to network losses, and the unreliable nature of eavesdropping (i.e., the sender does not expect eavesdropping nodes to acknowledge the transmission), the cost assigned to each edge is only an estimate. Different nodes may have different cost estimates for the same network edge. LMST is not designed to operate under these conditions, and does not guarantee connectivity unless all nodes share the same weighted graph. One might imagine solving this problem by having all nodes periodically share their edge weight estimates with their neighbors. However, one-hop agreement is not sufficient. In order to guarantee that the LMST preserves connectivity, all nodes who can see an edge must agree on its weight. On a unit disk graph, two nodes observing the same edge are maximally two hops away, however in an asymmetrical network like we observe in our testbed (recall Figure 3), they may be even farther. It is certain that performing per-edge consensus between nodes that may be an arbitrary number of hops from each other is considerably more expensive than the original CONCH algorithm. Also consider that periodically sharing these estimates would limit replanning to the same period, hampering the ability of DCONCH to adapt to changing conditions, thus negating one of its main advantages.

In order to mitigate this problem, we must first identify edges that have a high risk of being contentious, i.e., that may significantly impact the network topology based on small differences in cost estimation between nodes (see Figure 8). An edge $\overline{tv}$ considered potentially contentious if a node u estimates the cost on $\overline{tv}$ to be similar to an edge between a node adjacent to that edge and itself, such as $\overline{uv}$. In this case, it is possible that both u and t believe that the other node is responsible for connecting v to the network. We proceed by estimating the maximum possible divergence between edge cost estimates. Thus, we consider an edge $\overline{vt}$ to be contentious if and only if, for any node u in range of both v and t

$$|cost_u(u,v) - cost_u(v,t)| < \sum_t (1 - qos(u,v))$$

where $cost_u(v,t)$ is the cost of the edge $\overline{vt}$ as estimated by u, and $qos(u,v)$ is the link quality estimate at u for v (i.e., it represents the probability that v will send a message not heard by u, see Section III-A2 for details).

This problem is solved in DCONCH by forcing contentious edges to be included in every node's plan. This guarantees connectivity at the expense of redundant edges, (i.e., a poorer approximation of the actual MST).

V. RESULTS

In this section we present the results obtained in our experiments. First, we give simulation results over the data obtained during our real-world deployments. Note that these results estimate power consumption using the datalogger's power profile described in Section III-D. We proceed to describe the outcome of our testbed experiments, and conclude with a discussion of our results. A performance summary is provided in Table I, listing the percentage of measured data that was suppressed by each algorithm, along with the total energy

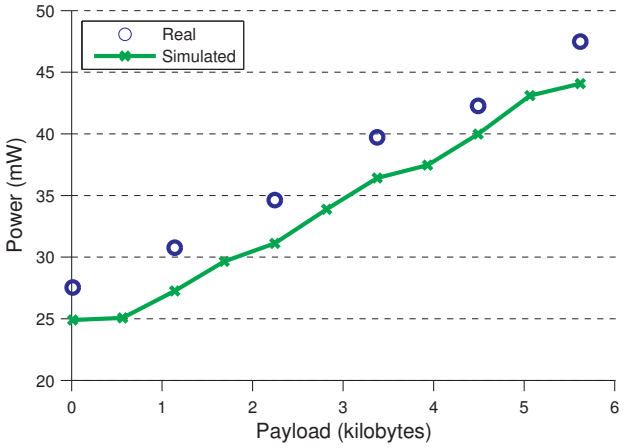

Fig. 9. Comparison of real-world versus simulated power consumption as a function of per-frame payload. Each data point represents the power consumed during one entire radio cycle (two minutes), averaged over a 14 minute period.

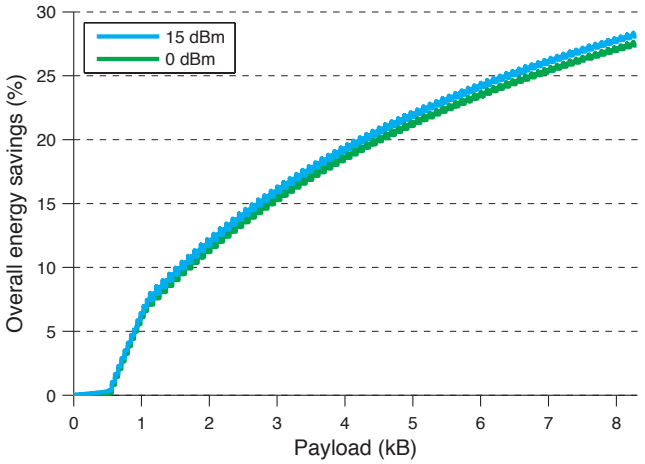

Fig. 10. Modeled effect of suppressing half of all sensor measurements on power saved, as a function of the total number of bytes to transmit per frame. We observe a slow increase until the minimum frame length is reached (four seconds), followed by periodic jumps that correspond to the need for additional packets.

Method	Algorithm	Suppression	Tx Energy	Error
Testbed	Default	0.0 %		0.0 °C
	CONCH	45.3 %		0.21 °C
Simulation	Default	0.0 %	1.38 mJ	0.0 °C
	CONCH	51.1 %	0.66 mJ	0.30 °C
	DCONCH	32.3 %	0.98 mJ	0.39 °C
	AR-CONCH	57.2 %	0.62 mJ	0.29 °C
	AR-DCONCH	63.9 %	0.51 mJ	0.33 °C

TABLE I
ALGORITHM PERFORMANCE

spent transmitting packets and the mean error of the resulting temperature estimate.

A. Simulation

During the real-world deployments, we logged two months of sensor data by SD card. This can be considered as ground truth temperature and neighborhood data, and thus used in conjunction with TOSSIM to perform realistic simulations.

1) Validation: In order to demonstrate the usefulness of our simulation environment, we compare the impact of increasing sensor payload on the power consumption of a real-world and simulated station. In Figure 9, we see the power consumed by the datalogger as the payload increases from 0 to 5 kB per radio frame. The monitored station is one hop from the sink, with no other network neighbors. We observe a strong match between simulated and real-world performance, and thus proceed to use our simulation environment to study the performance of each algorithm in detail below.

2) Performance: The default Sensorscope algorithm gives an average power consumption of 25.50 mW over our two month dataset. CONCH and AR-CONCH reduce the number of reported measurements by 51.1% and 57.2% respectively. DCONCH shows considerably lower performance than CONCH on our two month dataset due to redundant monitored edges created by our spanning tree approximation. However, AR-DCONCH is able to take advantage of fast replanning times to achieve a high rate of spatial suppression, overcoming the performance of AR-CONCH with a suppression rate of 63.9%. Ultimately, despite high rates of suppression, due to various sources of overhead inherent to the Sensorscope system, no algorithm shows a significant difference in overall energy consumption as compared to the default algorithm.

B. Testbed Results

1) Default Algorithm: We performed a ten day deployment using the original Sensorscope algorithm on our outdoor testbed near the end of March 2012. We observed high reliability, with 100% of all temperature measurements correctly logged to the onboard SD card, and only two sensor packets dropped over the ten day period (out of nearly 100,000). The dataloggers consumed 19.81 mW on average ($\sigma = 6.44$ mW).

2) Constraint Chaining: We performed a four week CONCH deployment during the month of April 2012. However, the network suffered from severe connectivity issues due to rain, causing all but one station to be partitioned from the sink for several days. This resulted in nodes almost always existing as direct reporters, as no links were deemed reliable enough over the five hour replanning period to become monitored edges. Additionally, due to implementation issues, our planning framework was not robust enough to recover from this extended period of unreliability, and thus did not function correctly near the end of the month when the rain had stopped.

Regardless, we extracted five days of useful data from this experiment. During this time we observed a 45.3% reduction in the number of sensor measurements sent, with an average temperature error of 0.21 °C ($\sigma = 0.45$ °C). Note that the temperature error is tied to the reporting threshold as described in Section IV-A (0.30 °C in our case). While our result is agreeable with previous simulation results for this algorithm, we observe that the average datalogger power consumption is similar to that of the original algorithm ($\mu = 23.21$ mW, $\sigma = 4.84$ mW). The slight increase in power consumption

is likely due to stations losing network connectivity and subsequently disabling radio duty cycling, permanently turning on their radios while they wait to receive a beacon.

C. Discussion

1) Overall Savings: It is clear from our results that the message suppression brought by the above algorithms has little effect on the Sensorscope datalogger's power consumption. We performed simulations in MATLAB in order to determine at what point the algorithmic choice has a significant impact on station lifetime. We precisely modeled the Sensorscope network frame (as described in Section III-A2), using values for power consumption, station density, and packet loss as measured during our outdoor deployments. Note that this model approximates average performance, and does not account for the bursty nature of unacknowledged packets.

Figure 10 shows that suppressing sensor data has an insignificant effect unless the station is transmitting a large amount of data per frame (i.e., more than about 2 kB), while the busiest node in our deployment was only responsible for about 30 bytes per frame. While transmitting is indeed considerably more expensive than other station activities (recall Figure 4), the datalogger still uses a significant amount of power even while the radio is off. Once this overhead is surpassed by increasing demands on the radio, message suppression begins to have a significant effect. We obtain a maximum overall power reduction of nearly 30% at the maximum frame length. Note that minimum versus maximum transmission power makes only a small difference, as radio receive durations during CSMA backoffs dominate the radio's power budget.

Note that this plot will vary depending on the aforementioned factors. As the network becomes more dense and packet loss increases, the positive effect of packet suppression is increased.

2) DConch Performance: We found that in our experiments DCONCH monitored on average 1.42 times as many edges compared to CONCH. Due to its ability to replan locally (i.e., more frequently), the spatial suppression rate in DCONCH is significantly higher (28.8% versus 12.9%). However, the increase is not enough to compensate for the extra monitored edges.

The effectiveness of this approach is visualized in Figure 11. We observe that, in the absence of overhead, the performance of the differential constraint drops off very quickly as the replanning period increases. A more powerful edge constraint must be used to leverage DCONCH's ability to replan more frequently. The AR model constraint benefits greatly from reduced replanning time, yielding results more favorable to DCONCH by lessening the impact of redundant edges and increasing the suppression rate overall. Ultimately, this change results in AR-DCONCH sending 10.5% fewer messages than AR-CONCH in our simulations.

These results could be further improved if a model of the exact process under measure is known. Any temporal model could potentially be used as an edge constraint as long as the network nodes have the requisite processing power.

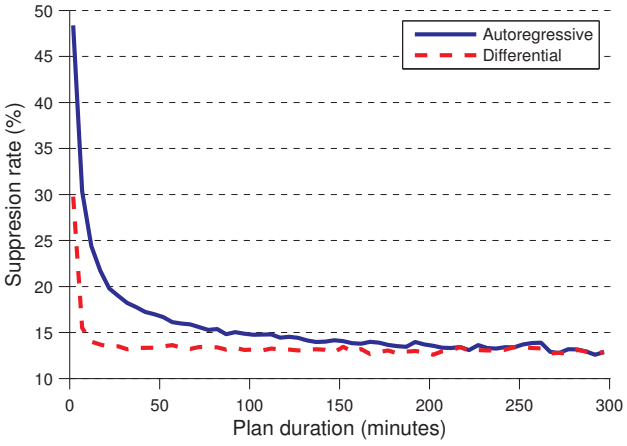

Fig. 11. Performance with respect to plan duration for both the AR model and original differential edge constraints. We observe that the AR model constraint is able to take greater advantage of high replanning frequencies. Note that this plot ignores the replanning overhead, i.e., the message cost of flooding the network with a new plan.

VI. Conclusions and Future Work

During the course of this project we took a suppression-based sensor network monitoring algorithm from literature and implemented it on a real system. We modified the algorithm significantly in order to make it feasible on real hardware. We found that it showed poor performance under the dynamic conditions typical in sensor networks, and proposed a fully distributed approach that is better able to adapt to unstable network topologies and changing environmental conditions. Finally, we used a custom-developed power monitoring board to show that even though these algorithms greatly reduce network traffic, they have no effect on Sensorscope's overall power consumption.

It is clear from our results that Sensorscope's current network architecture does not benefit from suppression-based approaches to efficient monitoring. Many ultra low-power network architectures have been proposed, however they often have extreme data latencies (e.g., Koala [25], Dozer [26]) or heavily rely on the typical periodic nature of environmental sampling (e.g., DISSense [27]). A low power protocol that can take advantage of variable amounts of data per cycle must be identified before we can observe any real-world gains.

While in this work we studied algorithmic performance on a typical sensor at its typical sampling rate (as configured by Sensorscope), performance estimates must be tied to the characteristics of the underlying environmental process before they can be applied more generally. Consider that the more severely a process is oversampled, the better suppression-based algorithms will appear to perform. There exists a body of literature that explicitly examines environmental relationships in the context of sensor networks (e.g., [28], [29]), and in the future we plan to use these relationships as a predictor of

algorithmic performance.

ACKNOWLEDGMENTS

This work was partially funded by "The Swiss Experiment" supported by the Competence Center Environment and Sustainability of the ETH Domain (CCES), and by the NCCR Transfer project "Ultra low-power wireless camera network for agriculture monitoring and pest attack detection" sponsored by the Swiss National Science Foundation.

REFERENCES

[1] A. Silberstein, R. Braynard, and J. Yang, "Constraint Chaining: on energy-efficient continuous monitoring in sensor networks," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, Jun. 2006, pp. 157–168.

[2] D. Tulone and S. Madden, "PAQ: Time series forecasting for approximate query answering in sensor networks," in *Proceedings of the 3rd European Conference on Wireless Sensor Networks (EWSN)*, 2006, pp. 21–37.

[3] ——, "An energy-efficient querying framework in sensor networks for detecting node similarities," in *Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, 2006, pp. 191–300.

[4] A. Prorok, C. Cianci, and A. Martinoli, "Towards optimally efficient field estimation with threshold-based pruning in real robotic sensor networks," in *IEEE Intenational Conference on Robotics and Automation*, May 2010, pp. 5453–5459.

[5] T. Arici and Y. Altunbasak, "Adaptive sensing for environment monitoring using wireless sensor networks," in *IEEE Wireless Communications and Networking Conference*, Mar. 2004, pp. 2347–2352.

[6] G. Quer, R. Masiero, G. Pillonetto, M. Rossi, and M. Zorzi, "Sensing, compression and recovery for WSNs: Sparse signal modeling and monitoring framework," *IEEE Trans. Wireless Commun.*, vol. 11, no. 10, pp. 3447–3461, 2012.

[7] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, Mar. 2005.

[8] X. Yang, H. B. Lim, T. M. Özsu, and K. L. Tan, "In-network execution of monitoring queries in sensor networks," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, 2007, pp. 521–532.

[9] D. F. Nadeau, W. Brutsaert, M. B. Parlange, E. Bou-Zeid, G. Barrenetxea, O. Couach, M.-O. Boldi, J. S. Selker, and M. Vetterli, "Estimation of urban sensible heat flux using a dense wireless network of observations," *Environ. Fluid Mech.*, vol. 9, pp. 635–653, 2009.

[10] S. Simoni, S. Padoan, D. Nadeau, M. Diebold, A. M. Porporato, G. Barrenetxea, F. Ingelrest, M. Vetterli, and M. Parlange, "Hydrologic response of an alpine watershed: Application of a meteorological wireless sensor network to understand streamflow generation," *Water Resources Research*, vol. 47, p. W10524, 2011.

[11] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, "The hitchhiker's guide to successful wireless sensor network deployments," in *Proceedings of the 6th ACM conference on Embedded Network Sensor Systems*, Nov. 2008, pp. 43–56.

[12] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope: Application-specific sensor network for environmental monitoring," *ACM Transactions on Sensor Networks*, vol. 6, no. 2, pp. 17:1–17:32, Mar. 2010.

[13] H. Dubois-Ferrière, L. Fabre, R. Meier, and P. Metrailler, "TinyNode: A comprehensive platform for wireless sensor network applications," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, Apr. 2006, pp. 358–365.

[14] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, Nov. 2003, pp. 126–137.

[15] W. C. Evans, A. Bahr, and A. Martinoli, "A flexible in situ power monitoring unit for environmental sensor networks," in *IEEE/RSJ IROS Workshop on Environmental Monitoring (WREM)*, Oct. 2012.

[16] ——, "Evaluating efficient data collection algorithms for environmental sensor networks," in *Proceendings of the 10th International Symposium on Distributed Autonomous Robotic Systems*, Nov. 2010, Springer Tracts in Advanced Robotics (2013). Vol. 83, pp. 77–89.

[17] J. Durbin, "The fitting of time series models," *Rev. Inst. Int. Stat.*, vol. 28, pp. 233–243, 1960.

[18] P. Brockwell and R. Davis, *Introduction to time series and forecasting*. Springer, 1994.

[19] R. Gallager, P. Humblet, and P. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and Systems*, vol. 5, no. 1, pp. 66–77, Jan. 1983.

[20] M. Elkin, "A faster distributed protocol for constructing a minimum spanning tree," in *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Jan. 2004, pp. 359–368.

[21] M. Khan, G. Pandurangan, and V. S. A. Kumar, "A simple randomized scheme for constructing low-weight k-connected spanning subgraphs with applications to distributed algorithms," *Theoretical Computer Science (Elsevier)*, vol. 385, no. 1–3, pp. 101–114, Oct. 2007.

[22] N. Li, J. C. Hou, and L. Sha, "Design and analysis of an MST-based topology control algorithm," *IEEE Trans. Wireless Commun.*, vol. 4, no. 3, pp. 1195–1206, May 2005.

[23] J. Cartigny, F. Ingelrest, D. Simplot-Ryl, and I. Stojmenović, "Localized LMST and RNG based minimum-energy broadcast protocols in ad hoc networks," *Ad hoc Networks (Elsevier)*, vol. 3, no. 1, pp. 1–16, Jan. 2003.

[24] F. J. Ovalle-Martínez, I. Stojmenović, F. García-Nocetti, and J. Solano-González, "Finding minimum transmission radii for preserving connectivity and constructing minimal spanning trees in ad hoc and sensor networks," *Journal of Parallel and Distributed Computing (Elsevier)*, vol. 65, no. 2, pp. 132–141, Feb. 2005.

[25] R. Musaloiu-E., C.-J. M. Liang, and A. Terzis, "Koala: Ultra-low power data retrieval in wireless sensor networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2008, pp. 421–432.

[26] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: Ultra-low power data gathering in sensor networks," in *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2007, pp. 450–459.

[27] U. M. Colesanti, S. Santini, and A. Vitaletti, "DISSense: An adaptive ultralow-power communication protocol for wireless sensor networks," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, Jun. 2011, pp. 1–10.

[28] M. C. Vuran and Ö. B. Akan, "Spatio-temporal characteristics of point and field sources in wireless sensor networks," in *IEEE International Conference on Communications*, Jun. 2006, pp. 234–239.

[29] X. Dong and M. C. Vuran, "Spatio-temporal soil moisture measurement with wireless underground sensor networks," in *The 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop*, Jun. 2010, pp. 1–8.