Packet-Level Attestation (PLA): A Framework for In-Network Sensor Data Reliability

ABU RAIHAN M. KAMAL and CHRIS BLEAKLEY, University College Dublin
SIMON DOBSON, University of St Andrews

Wireless sensor networks (WSN) show enormous potential for collection and analysis of physical data in real-time. Many papers have proposed methods for improving the network reliability of WSNs. However, real WSN deployments show that sensor data-faults are very common. Several server-side data reliability techniques have been proposed to detect these faults and impute missing or erroneous data. Typically, these techniques reduce the lifetime of the network due to redundant data transmission, increase latency, and are computation and storage intensive. Herein, we propose Packet-Level Attestation (PLA), a novel framework for sensor data reliability assessment. It exploits the spatial correlation of data sensed at nearby sensors. The method does not incur additional transmission of control message between source and sink; instead, a verifier node sends a validation certificate as part of the regular data packet. PLA was implemented in TinyOS on TelosB motes and its performances was assessed. Simulations were performed to determine its scalability. It incurs only an overhead of 1.45% in terms of packets transmitted. Fault detection precision of the framework varied from 100% to 99.48%. Comparisons with existing methods for data reliability analysis showed a significant reduction in data transmission, prolonging the network lifetime.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Performance

Additional Key Words and Phrases: Wireless sensor networks, sensor data, sensor faults, data collection, routing, sensor network, data reliability

ACM Reference Format:
Kamal, A. R. M., Bleakley, C., and Dobson, S. 2013. Packet-level attestation (PLA): A framework for in-network sensor data reliability. ACM Trans. Sensor Netw. 9, 2, Article 19 (March 2013), 28 pages.
DOI: http://dx.doi.org/10.1145/2422966.2422976

1. INTRODUCTION

Wireless sensor networks (WSNs) have attracted significant academic and industry interest because of their potential to sense and interact with the physical environment over a wide area and in real time. Researchers have designed and deployed WSNs for a range of applications, including environmental monitoring [Ingelrest et al. 2010], habitat monitoring [Szewczyk et al. 2004], scientific exploration [Tolle et al. 2005], infrastructure protection [Xu et al. 2004], and health monitoring [Chipara et al. 2010]. It has been found that a substantial portion of the data collected in real deployments is, in fact, faulty [Sharma et al. 2010]. For example, only 49% of the data collected

This work is partially supported by the Science Foundation of Ireland under grant number 04/RPI/1544, "Secure and Predictable Pervasive Computing," and by the Higher Education Authority PRTLI4 under grant number R10891, "NEMBES: Networked Embedded Systems."
Authors' addresses: A. R. M. Kamal and C. Bleakley, School of Computer Science and Informatics, Complex and Adaptive Systems Laboratory, University College Dublin, Ireland; S. Dobson, School of Computer Science, University of St Andrews, U.K.; Corresponding author's (A. R. M. kamal) email: raihan@ucdconnect.ie.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2013 ACM 1550-4859/2013/03-ART19 $15.00
DOI: http://dx.doi.org/10.1145/2422966.2422976

in Tolle et al. [2005] was correct. In the Great Duck Island experiment, 3–60% of the data collected by each sensor was faulty. Similar statistics have been found in other datasets, such as SensorScope and Intel Lab [SesnorScope 2008]. Since WSN nodes are typically deployed in unattended and hard-to-reach areas, manual checking and maintenance of the nodes is time consuming and expensive. Clearly, improving the reliability of the data gathered by WSNs is key to commercializing WSN technology.

Most previous WSN reliability research has focused on improving the reliability of data in terms of improving packet transmission from the source node to the sink [Gnawali et al. 2009; Puccinelli and Haenggi 2010]. A small number of publications, including this one, consider the problem of data faults. A data fault occurs when a node performs a sensing task in an erroneous way, resulting in faulty data which deviates from a true representation of the physical phenomenon to be measured. Data-fault detection by checking for sudden, large temporal variations in the data statistics at a single sensor is fairly straightforward and has been proposed previously [Balzano and Nowak 2007]. Single-node checking is generally not effective in detecting slowly evolving drift errors, calibration errors, or noise. Data faults which manifest themselves in this way can be more accurately detected by comparing the data sensed over time by multiple nodes. However, this approach is much more challenging. Most previous algorithms proposed for assessing data reliability using spatial methods have been designed for server-side deployment, that is, execution at the sink [Sharma et al. 2010]. This approach has three main disadvantages. First, it requires transmission of redundant data to the sink for analysis. In most cases, radio communication is the greatest consumer of energy in sensor nodes [Zhao and Guibas 2004]. Hence, relaying measurements from redundant nodes to the sink for the purposes of data checking significantly reduces the lifetime of the network. Second, checking is performed off-line. Many techniques presume that data collection is complete before verifying them. Hence real-time applications cannot benefit from them. Third, the approach leads to unacceptably long delays when the application does not require data to be transmitted back to the sink, for example, when nodes need only communicate locally [Tavakoli et al. 2010].

Herein, we propose Packet-Level Attestation (PLA), a novel lightweight, in-network framework for data reliability assessment in WSNs. The algorithm is based on the concept of nominating nearby data verifier nodes for each node. Typically nodes within close physical proximity exhibit similar data statistics, which can be characterized by their mean ($\bar{x}$) and variance (σ^2). Observed phenomenon may change significantly with time, but the spatial correlation between two nearby nodes typically remains stable [Liu et al. 2007]. We exploit this spatial correlation as an integral part of the data collection process. When a node is queried, the data packet is routed via one suitable verifier node and the source data is compared with the same quantity sensed at the verifier node. The verifier node piggybacks a data attestation bit onto the data packet which it sends on to the sink. This attestation bit provides an indication of the data reliability at the destination node. A lightweight, that is, low computational complexity and low memory requirement, algorithm is proposed for data validation. Unlike server-based validation, the checking algorithm must be sufficiently lightweight so as to run on typical sensor nodes which are characterized by an 8/16-bit RISC micro-controller, a few kilobytes of RAM, and a few hundred kbps data rate.

Many WSN deployments inherently exhibit spatially correlated data which can be exploited to increase redundancy. Sensor data from real-world deployments, such as Intel Data [2004] and SesnorScope [2008], reveal that this form of redundancy is common in WSN applications. Some applications do not naturally exhibit sufficient redundancy to support PLA. In these cases, there exists a trade-off between accepting lower reliability and deploying additional nodes to support PLA. It is worth mentioning that the term

attestation conveys a different meaning in the area of trusted computing (TC). In TC, attestation implies reporting the status of computing hardware or software, often with the aid of cryptographic techniques [Brickell et al. 2004]. In our context, the term attestation is used in a more general way, implying certification based on data similarity.

To the best of our knowledge, PLA is the first work which proposes validation of sensor data by its peer nodes with piggybacking of certification in a regular data packet. We outline the further contributions of this article as follows.

—PLA is a generalized framework for data verification which works on top of any routing protocol or data-collection strategy. PLA does not impose any constraints or limitations on the network protocol stack. Hence, network reliability techniques are complementary to PLA.
—A low-complexity verifier node selection algorithm is proposed for storage-constrained sensors.
—PLA supports delayed attestation in the case that the verifier node is faulty.
—The performance of PLA is assessed in terms of message size and compared with previous proposals.
—The fault detection accuracy is assessed using both the verifier node selection algorithm and a random selection process.
—The results of implementation of the validation algorithm on TelosB nodes using TinyOS are presented.
—The scalability of PLA for large networks is assessed in simulation.

The rest of this article is organized as follows. Section 2 presents the problem statement along with the system assumptions. Definitions and notation are stated in Section 3. Details of the algorithms along with the statistical basics used in PLA are discussed in Section 4. In Section 5, implementation, simulation, and numerical analysis results are presented. Section 6 introduces related work. Finally, Section 7 concludes the work.

2. PROBLEM STATEMENT

A typical WSN deployment exhibits two types of faults [Guo et al. 2009].

Network fault. These faults are caused by the sudden death of a node, insufficient network coverage to send a packet, packet loss due to routing failure, and so on. Network faults have been extensively studied by the research community. Both distributed [Marti et al. 2000] and centralized [Staddon et al. 2002] algorithms have been proposed to improve reliability in the event of network faults.

Data fault. In the case of a data fault, a node performs all sensing functions but the task is performed in an erroneous way, resulting in data which does not accurately represent the true physical phenomenon being propagated through the WSN system. Data faults can be classified in various ways. A comprehensive study of sensor faults along with a systematic taxonomy was provided in Ni et al. [2009]. In this work, we follow the general classification of sensor faults given in [Sharma et al. 2010].

—*Constant*. The node reports a constant value for a long duration. The reported value is generally outside the normal range. During the fault, the data has a standard deviation σ of 0.
—*Short*. There is a sudden change in sensor readings between two consecutive readings. Short faults are easier to detect than others due to the presence of a sudden irregular high/low spike.

—*Noise*. Several successive readings have a small error. Noise faults result in a sudden change in the data's variance. They can be modeled as the occurrence of several, consecutive Short faults.
—*Calibration*. Sensor data may have offsets or have incorrect gain. Both offset and gain may change with time, causing drift faults.

The problem addressed herein is accurate data-fault detection in a query-based, multihop WSN subject to the following constraints.

—The energy consumption of the solution should be minimized so as to allow a long network lifetime.
—The solution should be lightweight so as to allow implementation on typical WSN micro-controllers.
—The solution should be efficiently scalable to allow implementation on WSNs comprised of at least 100 nodes.
—The solution should ensure high fault-detection accuracy with insignificant delay overhead.

A query-based WSN is defined as a WSN in which a subset of the nodes transmit data to the sink at regular intervals for a period of time according to a user-initiated query. A multihop WSN is defined as a WSN in which data is transmitted to the sink via a number of intermediate battery-powered nodes. Query-based multihop WSNs are common in many applications [Corke et al. 2010].

The following assumptions are made.

(1) It is assumed that the physical quantity being sensed is spatially diffused and that the density of sensor placement is sufficient to ensure that there is spatial correlation between the sensed data at different nodes with close physical distance and environment. This is reasonable for local area WSNs recording quantities such as temperature, humidity, pressure, light, etc.
(2) It is assumed that in the initialization phase, most sensors are non-faulty. This is justifiable since most sensor faults are the result of battery power drain, sensor aging, and operational wear and tear.
(3) There is local synchronization between a node and its one-hop neighbors. Since node failure, link failure, and traffic congestion occur very frequently, maintaining a global clock is very hard to achieve. A local clock synchronization is assumed with an accuracy better than the rate of change of the physical quantity being sensed.
(4) It is assumed that there is a suitable routing technique for routing data from source to sink. Formally, there is a path $S_0S_1S_2S_3 \ldots S_n$, where S_0 and S_n are the source and sink, respectively. The proposed method does not incur any change in routing except that the path starts at S_1 instead of S_0.
(5) The network topology is static and known in advance. Nevertheless, the proposed framework can operate under conditions of a relatively dynamic topology.
(6) It is assumed that there is no security threat from an external attacker which can cause node misbehavior.

3. DEFINITIONS AND NOTATION

Following are definitions of the terms used in the rest of the article.

—*Source Node*. The source node is any node within the network that can be queried. In the context of a particular query, the source node is denoted as the ith node, N_i. The source node sends its sensed data to the sink node.

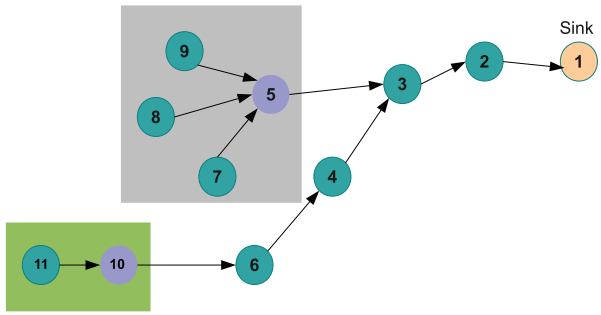

Fig. 1. Example WSN topology for PLA.

—*Query*. A query is a request for a series of data values from the sink to a particular source node. The sampling rate and duration of data collection is specified. The WSN can process multiple queries simultaneously.

—*Sink Node*. The sink node initiates queries to the network and receives the data response from the network. There must be at least one, and may be more than one, sink node in the network.

—*Neighbor Nodes.* Neighbor nodes are within one hop of a particular node. The term $Neigh(N_i, Null)$ denotes the subset of nodes that are one hop from node N_i. The term $Neigh(N_i, Sink)$ denotes the subset of one hop neighbors of N_i which have a lower or equal hop count to the sink node. Thus, $Neigh(N_i, Sink) \subseteq Neigh(N_i, Null)$.

—*Verifier Node.* A verifier node checks and attests for the sensed data transmitted by the source node. Nodes can simultaneously be source nodes for one query and verifier nodes for another query. A verifier node may verify data from one particular source node which is termed as *one-one configuration* or simply *1-1*. On the other hand, a verifier node may simultaneously verify multiple source nodes' data, resulting in a *many-one configuration* (*m-1*). A network can combine both configurations, depending on the topology. Data gathered for the purposes of verification is not sent to the sink. Tha verifier node of N_i is denoted by $V(Ni)$. Note that $V(N_i)$ cannot be N_i itself, that is, $V(N_i) \neq N_i$. $V(N_i)$ must be a member of $Neigh(N_i, Sink)$.

—*Sensor Data.* Sensor data D is represented as a matrix of $N \times T$ dimension, where N and T correspond to sensor index and sensing time, respectively.

$$D = \begin{bmatrix} d_{11} & d_{12} & d_{13} & \dots & d_{1t} \\ d_{21} & d_{22} & d_{23} & \dots & d_{2t} \\ d_{31} & d_{32} & d_{33} & \dots & d_{3t} \\ \dots & \dots & \dots & & \\ d_{n1} & d_{n2} & d_{n3} & \dots & d_{nt} \end{bmatrix}.$$

Each row of D contains the data for an individual sensor; each column contains the data from all sensors at the same point in time. Note that the WSN does not sense the complete data record D. Only the portions of D queried are transferred to the sink.

—*Monitoring.* In monitoring applications, the WSN measures the physical environment on a regular, ongoing basis in order to obtain a record of conditions.

An example topology is shown in Figure 1. In this example, node 1 is the sink. Any node in this network (from node 2 to 11) can provide sensory data to the sink and hence are source nodes. The neighbors of node 7, for instance, are nodes 4, 5, 8, and 6. Of

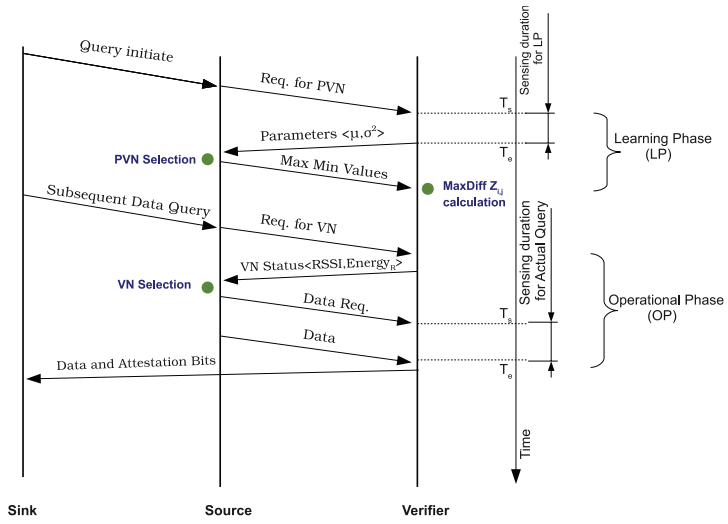

Fig. 2. Message sequence diagram for the PLA.

these, nodes 4 and 5 are closer (less hop count) to the sink and so could be verifier nodes, while the other two can not. So, nodes 4 and 5 are possible verifier nodes. Among them, node 5 is selected as a verifier node for node 7 based on its data statistics, link quality, and residual energy. Here, node 5 maintains many-to-one configuration as it verifies nodes 7, 8, 9 simultaneously, while node 10 retains its one-to-one configuration.

That is,
$Neigh(7, Null) = 4, 5, 8, 6,$
$Neigh(7, 1) = 4, 5,$
$V(7) = V(8) = V(9) = 5,$
$V(11) = 10.$

4. PROPOSED PLA FRAMEWORK

The operation of the PLA framework can be summarized as follows.

During a learning phase (LP), the statistical relationship between the data sensed at different nodes is established. Based on analysis of these relationships, each node is allocated a number of possible verifier nodes (PVNs) with similar data statistics. During the operational phase (OP), when a node's value is queried, it first selects one verifier node from its PVNs. The source node then sends a data packet to the verifier node. When the verifier node receives the data packet, it compares its own sensed data and the received data. Based on the previously established data relationship, the verifier node determines if the source data is consistent with its own data. A data attestation bit indicating consistency is added to the data packet which is forwarded toward the sink. The sink may take remedial action based on the consistency of attestation. Remedial action is application dependent and may include querying other nodes or manually replacing the faulty node. Overall message sequence for the PLA framework is shown in Figure 2.

The LP and OP are described in more detail in the following sections.

Table I. Request Message Format

Field	Meaning
T_i	Interval of periodic sensing
T_p	Total sensing period

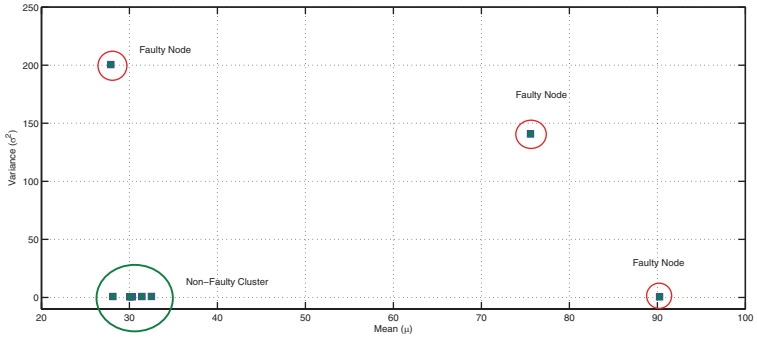

Fig. 3. Scatter plot of dataset statistics [Lim and Bleakley 2010].

4.1. Learning Phase (LP)

Overview. The LP is initiated by the sink and takes place before data is passed to the application. Although the process is triggered by the sink, the entire process is distributed. This significantly reduces energy consumption compared to a centralized server-side approach, since data is only exchanged locally and does not need to be transmitted back to the sink for processing.

At the start of the LP, all source nodes N_i issue a request message to all of their one-hop neighbours $Neigh(N_i, Sink)$ to gather a sample dataset. The request message is defined in Table I. Upon receiving the request, each node $N_j \subseteq Neigh(N_i)$ starts sensing at a frequency of T_i Hz for a duration of T_p seconds. The source node itself collects data at the same rate and over the same period. Both T_i and T_p are set according to application requirements.

At the end of the sensing time, all one-hop neighbor nodes provide a statistical summary (mean and variance) of their data to the source node. Since only the essential parameters are transmitted by each neighbor the data traffic is low, which is desirable for resource-constrained sensor networks. The source compares the data summary to its own data statistics. Based on this, the source node selects a number of possible verifier nodes (PVNs). The PVNs are one-hop neighbors which display similar data statistics to the source and so can be used to check the source's data. After PVN selection, the source node broadcasts its extreme data values (minimum d_{min}, maximum d_{max}). These values are used to validate data during the operational phase.

Next we describe PVN selection and data attestation details.

PVN Selection. The goal of PVN selection is to identify nodes which will be effective checkers for the source node. Statistical metrics are used here to identify a set of possible verifier nodes (PVNs). Considering the property of spatial correlation among sensor data collected from nearby sensors, statistical metrics such as mean and variance (or standard deviation) can be effectively utilized for identifying similar data trends. Two datasets with different means may have similar variance. Again, datasets may have dissimilarities in their variance but identical mean. Therefore, it is essential to check both parameters jointly. So, we postulate that the mean and variance of sensor data

records can be used to identify nodes that can serve as effective verifier nodes for any given source node. the fact is illustrated in Figure 3. Mean and variance are used to identify non-faulty data and ot reject faulty data.

For the sake of completeness of the work, statistical measures used in PLA are presented here before describing the selection algorithm.

Mean ($\bar{x}$). If the observations in a sample of size w are $x_1, x_2, x_3, \ldots x_w$, then the sample mean is defined as follows.

$$\bar{x} = \frac{\sum_{i=1}^{w} x_i}{w}. \tag{1}$$

Variance (σ^2). If $x_1, x_2, \ldots x_w$ is a sample of w observed data points, then its variance is

$$\sigma^2 = \frac{\sum_{i=1}^{w} (x_i - \bar{x})^2}{w-1}. \tag{2}$$

Variance (σ^2) is used rather than standard deviation (σ) since it avoids calculation of the square root. The parameters ($\bar{x}$, σ) and ($\bar{x}$, σ^2) are identical from a statistical point of view.

Effective Estimation of σ^2. Equation (2) suggests that the computation of σ^2 requires calculation of $\bar{x}$. This would require storage of w data values prior to calculation of the variance. Due to tight memory budget of sensor nodes, local storage is a major concern in WSN protocol and application development. A more efficient computation is used here which obviates the local storage of w values. It is derived as follows.

$$\begin{aligned} \sigma^2 &= \frac{\sum_{i=1}^{w} (x_i - \bar{x})^2}{w-1} \\ &= \frac{\sum_{i=1}^{w} (x_i^2 + \bar{x}^2 - 2x_i\bar{x})}{w-1} \\ &= \frac{\sum_{i=1}^{w} x_i^2 + w\bar{x}^2 - 2\bar{x}\sum_{i=1}^{w} x_i}{w-1}. \end{aligned} \tag{3}$$

Since $\bar{x} = (\frac{1}{w})\sum_{i=1}^{w} x_i$, Equation (3) reduces to the following form.

$$\sigma^2 = \frac{\sum_{i=1}^{w} x_i^2 - \frac{(\sum_{i=1}^{w} x_i)^2}{w}}{w-1}. \tag{4}$$

It is important to note that calculation of σ^2 according to Equation (4) requires storage of only three variables. Equations (2) and (4) produce identical results.

PVN Selection Algorithm. PVN selection is performed by clustering the nodes according to their dataset statistics. This is done by finding the most commonly occurring mean and variance values using a histogram method. The algorithm is formally explained in Algorithm 1 and the parameters are listed in Table II.

The range of mean values reported by the sensors is subdivided into a number of buckets. Each bucket covers a unique range of values. All buckets have the same width. The total number of buckets and the width of the buckets is application specific and is set a priori. The width of a bucket should be small enough so that erroneous readings are not accommodated within the same bucket as a normal reading. Too small a width may result in too few PVNs. The algorithm proceeds by counting the number of nodes with mean values occurring in each bucket. All one-hop neighbor nodes and the source node are included in this count. In addition to storing the count in each bin, the

Table II. Parameters for PVN

Notation	Meaning
$X\ Bucket_i$	Ith bucket for a particular type here Val(X): M for μ and V for σ^2
$X\ Bucket_i.f$	A number indicating frequency of datasets belonging to this bucket
$X\ Bucket_i.members$	A list of nodes contributing for its frequency
$X\ Bucket_i.add(N_j)$	Ith node is added to its member list
$X\ Bucket_i.InRange(X)$	Returns true if X lies within this subinterval

algorithm also stores the node IDs associated with each bin. The process its repeated using the variance values, although the buckets are different.

The algorithm then searches the histograms for the buckets containing the most frequently occurring mean and variance values. The node IDs corresponding to these buckets are retrieved. The nodes which occur in both buckets (mean and variance) make up the PVN list. The source node stores the PVN list until the next round of the PVN selection is performed. The next round of the selection process is performed after the expiration of its predefined lifetime T_{next}. T_{next} is application specific but is typically in the range of weeks or months.

If the source node's data statistics belong to a different bucket than those of the PVNs, then the algorithm returns FALSE, indicating that the source itself is faulty. In this case, a special control message is sent to the sink. The sink reissues the data collection request to a different node which repeats the entire process.

Calculation of MaxDiff $Z_{i,j}$. For each PVN, the source calculates the maximum differences between the source dataset and the PVN datasets. The source sends two extreme values sensed during the PVN selection process to each PVN. Each PVN uses its own extreme values obtained during the PVN selection phase to calculate *MaxDiff* as follows.

$$Z_{i,j} = Max(d_{i,t}, d_{j,t}) - Min(d_{i,t}, d_{j,t}), \quad for \quad t = T_s \quad to \quad T_e,$$

where N_i and N_j are the source node and PVN, respectively, and subscripts s and e denote the starting and ending points of data sensing during LP.

$Z_{i,j}$ is the maximum difference between the data of two nearby sensors N_i and N_j. The data trend may vary but stable pairwise differences between neighboring nodes are usually maintained. In other words, we assume $Z_{i,j}$ can be effectively used by N_j to validate data transmitted by N_i for the entire lifetime of the network. We checked the validity of this assumption using our data trace as discussed in the evaluation section.

4.2. Operational Phase

Overview. Once the learning phase is complete, the network is almost ready for regular operation. During the LP, each PVN obtains its *MaxDiff* $Z_{i,j}$ value. Before data transmission begins, the source node selects its verifier node (VN) from the PVN list for a particular query. Apart from data similarity, the VN selection process also considers link quality and residual energy. The selected VN is locally synchronized with the source node and starts sensing simultaneously at the same sensing frequency and duration as the query made to the source. The source sends data packets to the VN which is on its routing path to the sink. The VN verifies each data value using

ALGORITHM 1: PVN Selection

Input variables: $Mean[1..N]$,$Variance[1..N]$
2 arrays representing mean and variance reported by several sensors
Output variables: *RESULT* a FIFO queue,a list of Possible Verifier Nodes, initialized to *NULL*
SOURCEDATA a boolean variable to indicate source data status, initialized to $FALSE$
Local Variables: **R**, no. of buckets
N, no. of sensors
MMAX,VMAX, maximum value for selecting the winning bucket
Initialize Variables:
initilize all f and *members* to 0 and Null
for $i = 1$ to N **do**
  for $j = 1$ to R **do**
    if $MBucket_j.InRange(Mean[i])$ **then**
      $MBucket_j.f++$;
      $MBucket_j.add(N_i)$;
      set $j = R$;
    end if
  end for
  for $k = 1$ to R **do**
    if $VBucket_k.InRange(Variance[i])$; **then**
      $VBucket_k.f++$;
      $VBucket_k.add(N_i)$;
      set $k = R$;
    end if
  end for
end for
// find the maximum
$MMAX = FindMax(MBucket_[1..R].f)$;
$VMAX = FindMax(VBucket_[1..R].f)$;
// check source data

if $MBucket_{MMAX}.InRange(Mean[source])$ **then**
  if $VBucket_{SMAX}.InRange(Variance[source])$ **then**
    $SOURCEDATA = TRUE$;
  end if;
end if;
$RESULT = (MBucket_{MMAX}.members) \cap (VBucket_{VMAX}.members)$;

its own sensed data and *MaxDiff* $Z_{i,j}$. Finally, a status bit is constituted reflecting the consistency of the source data and the VN's data.

VN Selection. VN selection is based on the following three parameters.

(1) *Data Similarity.* The VN is chosen from the list of possible VNs, obtained as explained in Section 4.1. This ensures that both the source node and the verifier node exhibit a similar data trend.
(2) *Link Quality.* Since the source node must send the data to the verifier node, the link between them should be reliable. Herein, the received signal strength indicator (RSSI) is selected as the predictor of link quality since it is platform independent [Srinivasan and Levis 2006]. It is important to note that the selection process can utilize any other equivalent link metrics, such as the link quality indicator (LQI).

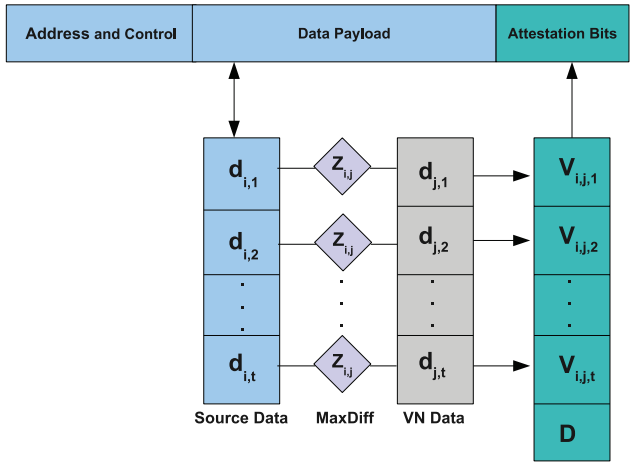

Fig. 4. VN inserts attestation bits.

(3) *Residual Energy.* The source node should ensure that the VN has sufficient energy to carry out the checking task. The VN piggybacks voltage level measurements on packet acknowledgments to the source node. This allows the source node to maintain an estimate of the remaining energy at the VN.

The source selects a member from the PVN list, offering the chance to work as a VN for the next session. The source node uses a round-robin schedule to select a VN from its PVN list. First, the source sends a request message to the one of the PVN members. Then the PVN member replies to the source. The reply message contains (i) an RSSI used to measure the link quality and (ii) the internal voltage level. Recent studies on RSSI and power consumption of sensor nodes reveal that RSSI greater than −87 dBm is considered a good link quality and voltage level greater than 2.4 Volts is an indicator of good residual energy for a typical sensor node [Puccinelli and Haenggi 2010; Deshpande et al. 2004]. Based on these standard values, the source node accepts or rejects the VN. If the criteria are not met, it selects the next member in the PVN list and the process is repeated. Maintenance of the PVN list is explained at the end of this section.

Data Attestation. After the source selects a VN, it sends a data request similar to that described in Table I. Sensing then proceeds simultaneously at the source node and VN. The source sends data packets to the VN. When a packet arrives, the VN also has its own readings corresponding to each data value in the incoming packet. VN also possesses its own *MaxDiff* value $Z_{i,j}$ which was computed during the LP. Due to the spatial correlation the absolute difference between any two readings from the source and VN (measured at the same time) should not exceed the $Z_{i,j}$ value. If this check is satisfied, the corresponding status bit is set to 0. Otherwise, it is set to 1, indicting that the source data and verifier data are not consistent. Using our notation,

$$V_{i,j,t} = \begin{cases} 0 & \text{if } \mid d_{i,t} - d_{j,t} \mid \leq Z_{i,j}, \\ 1 & \text{otherwise.} \end{cases}$$

The process is shown in Figure 4.

Finally, the attested data packet is routed to the destination node, usually the sink. The application running at the destination can inspect the attestation field and act accordingly.

The VN periodically sends feedback to the source node. The feedback contains its remaining energy level. Unlike link quality indicators, such as RSSI, which are often intermittent, the residual energy level is generally monotonically decreasing for battery-powered sensor nodes. PLA avoids per-packet acknowledgment. Instead, the VN sends acknowledgment to the source after it receives a certain number of data packets. This information is used by the source to select a VN from the PVN list.

Delayed Attestation, D Bit. Since the source node (SN) relies on the VN, it is essential to take measures to detect possible faults at the VN. Fortunately, faults in sensor data can be detected using their statistical properties [Sharma et al. 2010]. The VN computes its data variance within a fixed time window of length W. The length of W largely depends on the nature of the data and sampling rate. A general rule is that W should capture at least 50 samples. A value less than this fails to reflect the evolving nature of the data being monitored.

Whenever a VN finds its $\sigma^2 \longrightarrow 0$, it indicates a CONSTANT fault. An observation of $\sigma_i^2 = m \times \sigma_{i-1}^2$ suggests a NOISE fault, where m is a large multiplicative factor and σ_{i-1}^2 is the variance calculated for the previous window.

If the VN detects that it itself is faulty, then it uses its most recently sensed non-faulty window data for checking. It sets the D flag in the data packet to indicate *delayed attestation*. The VN notifies the SN, and the SN selects a new VN from the PVN list. The rationale of delayed attestation is that the most recently used data sample is not very stale and can effectively be used for checking the current data.

PVN List Maintenance. The effectiveness of PLA largely depends on the reliability of the VN. It is essential to maintain the PVN list via appropriate periodic updates. As described previously, each source builds its own PVN list during the LP. This section describes how a source node maintains this list during its valid lifetime T_{next}.

(a) *Member Switching.* The SN uses a non-prioritized round-robin schedule to ensure load balancing among its verifier nodes. Once a VN is selected, the source can switch to another VN in the following four cases.
   (i) The allocated time slot is finished so the next member from the PVN list is selected.
   (ii) The source node periodically receives acknowledgments from its current VN. It can switch to another VN if the VN's remaining energy level goes below the normal operating voltage.
   (iii) The source is informed by the VN whenever delayed attestation is performed. In this case, another VN is selected since the current VN is reported to be faulty.
   (iv) The SN maintains a link reputation vector (LRV) for each nearby node. The LRV records radio link quality measurements between the SN and VN. A bit is added to the LRV when a periodic acknowledgment is received from the associated node. The bit is set to 1 when the SN detects link quality above a predefined threshold L_{th}; otherwise, it is set to 0. The LRV acts as first-in-first-out queue of fixed length (typically 2 bytes). Thus it maintains the status of the links for the last 16 rounds. If the SN observes a 1 to 0 transition in the most recent bit of the LRV for the current VN, it deselects it and switches to another VN.

(b) *Member Addition.* Although PLA considers the network topology to be static, nodes may die for many reasons [Corke et al. 2010]. As a result, node repair and replacement are performed to ensure required sensing coverage and network connectivity. Whenever a node joins the network, the entire process of the learning phase is performed. Thus the new node may be added to a source's PVN list.

(c) *Member Removal.* A registered VN can be removed from the list permanently under the following two circumstances.

(i) As part of the D bit setting, the source switches to the next VN. However, the reportedly faulty VN is not deleted from the PVN list immediately; rather, it is given more chances, relying on the fact that faults often are sporadic. A member is only deleted from the list when the source finds it faulty more than R times consecutively. The value of R is dynamically adjusted accordingly to the length of the list (i.e., number of members). For shorter lists, a larger value is desirable and vice versa. This ensures that the faulty node is deleted only when it remains faulty for a significant period of time.

(ii) Every time the SN receives an acknowledgment, it updates the corresponding LRV accordingly. An LRV with mostly consecutive 0 entries indicates poor link for a prolonged duration. An LRV with a long sequence of alternating 0s and 1s implies that radio connectivity is intermittent. In both cases, the SN cancels the PVN membership of the corresponding node. At the same time, the SN clears all entries of the LRV specific to the VN. If a node later obtains good connectivity after being expelled from the list, it requests re-registration by the SN. Request control messages are sent at a predefined interval which is much longer than the regular sensing interval. The SN rebuilds the LRV after reception of each control message from the VN and re-registers it if the LVR is satisfactory.

(d) *Process Termination.* In extreme cases, the source may fail to select any suitable VN because either the PVN list is empty or no member of the PVN is eligible to be selected as VN. In this case, the verification process is terminated and members of PVN work as relay nodes. This ensures connectivity from source to sink if some nodes are faulty.

5. EVALUATION

This section describes the results of evaluation of the performance of the proposed algorithm. Performance was evaluated in three ways. First, a small testbed network was set up and the performance of the PLA was assessed in terms of power consumption and memory requirements. Second, the accuracy of the method in detecting data faults was assessed in numeric simulations. Third, the number of bytes transmitted using, PLA was calculated analytically. This was compared with two previous methods. Finally, the scalability of the method was assessed in network-level simulations.

5.1. Testbed Network

PLA was implemented in nesC running on TinyOS. nesC (network embedded systems C) is a component-based, event-driven language highly optimized for resource-constrained embedded devices. TinyOS is an open-source operating system targeted for wireless sensor networks [Tinyos 2010]. TinyOS version 2.1.1 and nesC Compiler version 1.3.1 were used. The performance of the network with PLA was compared to that of a simple relay network without PLA. Henceforth, we use the terms *relay node* and *nonPLA node* interchangeably in the remainaer of the article.

Network. The relevant portion of the network was implemented in the laboratory, this is, the source node, verifier node (and relay node), and basestation.

The Crossbow TelosB mote TPR2400 [Crossbow 2010] was used with the following attributes.

—8 MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash).
—250 kbps 2.4 GHz IEEE 802.15.4 Chipcon Wireless Transceiver.
—Integrated humidity, temperature, and light sensors.

Two network setups were established, as shown in Figure 5.

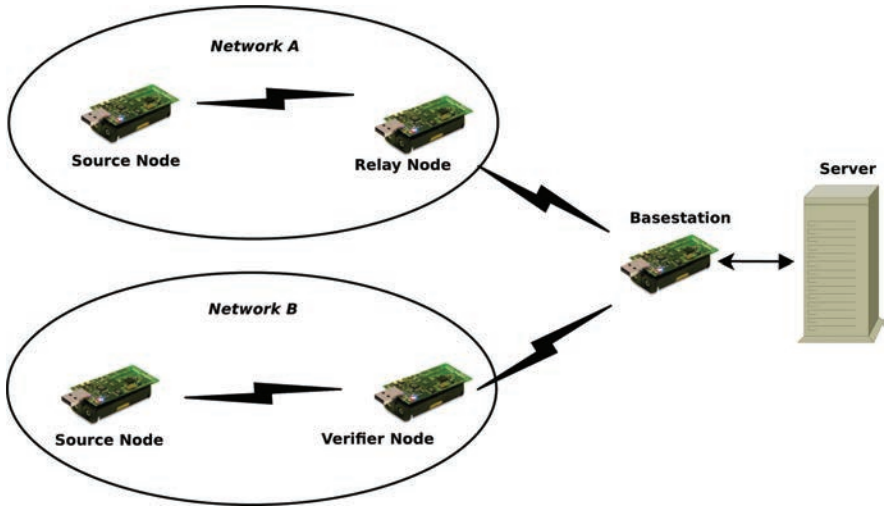

Fig. 5. Network setup.

—*Network A (without PLA)*. In this network, the source's one-hop neighbor receives packets from the source and simply forwards (relays) them to the basestation.
—*Network B (with PLA)*. In this network, the source's one-hop neighbor receives packets from the source, verifies them using PLA, and then forwards data and attestation information (in the same packet) to the basestation.

The physical distance between each peer sensor node was 20 meters (indoor).

The networks were configured so that the source would collect MAX_READINGS temperature measurements and then transmit the values in a single radio packet together with other useful information, such as packet sequence number, mote ID, and voltage level. The packet structure is the following.

```
typedef nx_struct RadioMsg{
nx_uint16_t moteid;
nx_uint32_t counter;
nx_uint16_t statusbits;
nx_uint16_t verifieraddr;
nx_uint16_t t_voltage;
nx_uint16_t t_value[MAX_READINGS];
}RadioMsg;
```

Program Size. One major problem with WSN protocol and application development is the inherent memory constraint of sensor nodes. Typically, a TelosB sensor has a RAM size of 10 KBytes. A node stores only a few variables to compute its mean μ, variance δ^2, maximum value, and minimum values. Since the number of one-hop neighbors (N) of the source node may vary, we assessed the program size for the PLA learning phase with different settings of N. The results are shown in Figure 6. Even with $N = 32$, the RAM footprint of PLA (LP) at the source node consumed only 8.13% of the available memory of a typical sensor.

The program sizes for a verifier node (PLA) and a normal relay node (without PLA) were also evaluated and are given in Table III. Note that the verifier node RAM and ROM size were only 5.6% and 2.1% higher than for a nonPLA relay node.

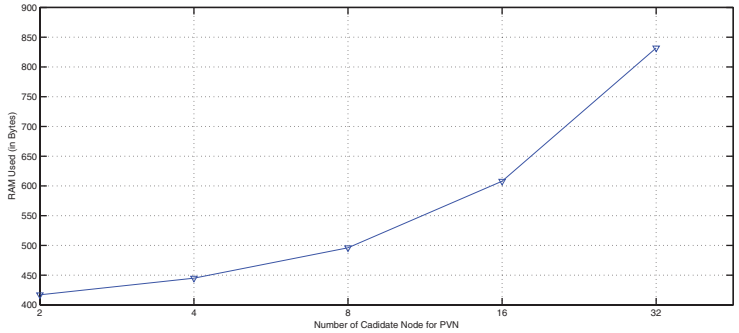

Fig. 6. Program size for LP.

Table III. PLA Program Size

	ROM (in Bytes)	Increase in %	RAM (in Bytes)	Increase in %
Relay Node	476	–	18,586	–
Verifier Node	504	5.6	18,976	2.1

Table IV. Node Lifetime for PLA

Packet Size	Networks	Lifetime (in Packets)	Lifetime (in Seconds)	Percentage (%)
MAX_READINGS = 15	Network A	1,037,200	726,040	100.00
	Network B (PLA)	1,028,900	720,230	99.19
MAX_READINGS = 31	Network A	497,856	696,998	100.00
	Network B (PLA)	487,401	682,361	97.9

Network Lifetime. Networks A and B were run independently with fresh batteries and temperature data were collected until they ran out of energy. The experiment ended once the internal voltage dropped below 2.4 V. The sensing sampling period T_i was set to 50 ms. In practical application deployments, the typical interval would be much larger than this value, but this value was used to shorten the total experiment time. After the lifetime of both networks expired, the collected data were analyzed to assess the impact of PLA, since two networks were set under identical distance and environmental factors.

In the experiments, MAX_READINGS = 15 and the length of the data payload was 42 bytes. PLA and nonPLA successfully reported a total of $1,028,900$ and $1,037,200$ packets, respectively, as stated in Table IV. The table also explains the results with a different setting of MAX_READINGS = 31. Network B reported 98.55% of the packets processed by Network A. Network B processed around 1.45% less packets due to energy consumption for both in-network computation and larger message transmission.

Energy consumption was assessed with the help of the internal voltage level of the relay node and verifier node in Network A and Network B. Figure 7 shows the variation of voltage level with time both for the verifier node and relay node. Clearly, the voltage level trends were almost identical for both nodes.

These results show that PLA incurs a negligible penalty in terms of the lifetime of sensor nodes.

5.2. Accuracy

This section details the accuracy of the PLA framework in two stages. First, the verification accuracy was assessed. Second, the correctness of the PVN selection algorithm was tested. In both cases, the results were obtained by numeric simulation in Matlab.

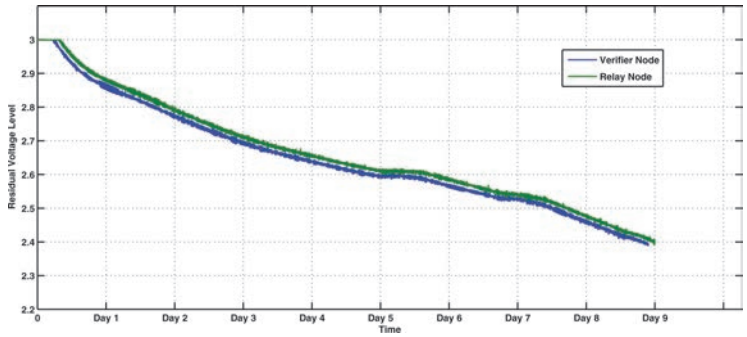

Fig. 7. Comparative power consumption.

Table V. Data Statistics

Field	Value (Source Data)	Value (Verifier Data)
Number of data points	1, 5433, 500	15, 433, 500
Sampling Interval	50 ms	50 ms
Maximum Value	6, 481.00	6, 642.00
Minimum Value	5, 838.00	5, 817.00
Mean Value	6, 027.77	5, 986.64
MaxDiff Zi, j	107	107

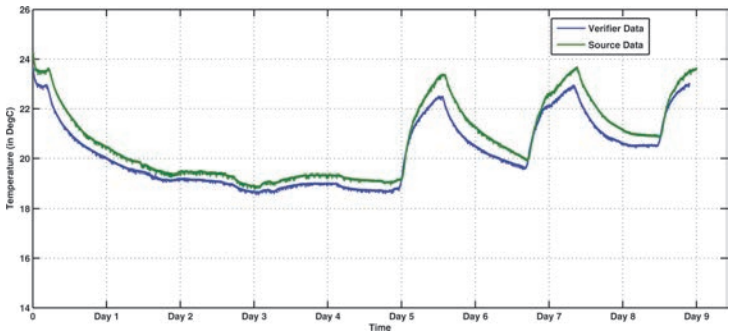

Fig. 8. Stability of spatial correlation.

5.2.1. Verification Accuracy. The accuracy of the proposed algorithm was assessed by injecting faults into a dataset containing temperature measurements from a real deployment, as described in the previous section. The faults were generated according to the fault model described next. Two scenarios were considered—faults at the source and faults at the source and verifier.

Dataset. Two WSN nodes were configured to collect temperature data periodically in a laboratory setting. The nodes were set 20 meters apart. The recorded dataset characteristics are listed in Table V. Listed values are presented in the raw sensed format without conversion.

Figure 8 shows the variation in temperature measurements at the source and verifier nodes with time. Figure 9 shows that the data has a small absolute difference, that is, $|di, t\text{-}d_{j,t}|$. The first 500 readings from both nodes were used in the learning phase to calculate the MaxDiff value, $Z_{i,j}$, which is also listed in Table V.

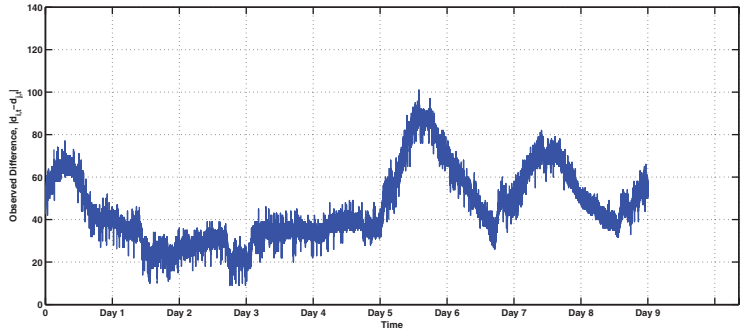

Fig. 9. Stability of spatial correlation, relative difference.

ALGORITHM 2: :*FaultInsertion*

Input: $d_{i,t}$ for t=1 to n, Input time-series data
P, Q, R, total faults to be inserted of type SHORT,CONSTANT,NOISE namely
Output:$d_{i,t}$, time-series sensor data with a total of $P + Q + R$ faults
(for SHORT faults:)

for $j = 1$ to P **do**
  $k = Random(1 : n)$;
  $f = random(.1 : 10)$;
  $d'_{i,k} = d_{i,k} + d_{i,k} \times f$;
end for
(for CONSTANT faults:)
$l = Random(1 : n)$;
$X = random(10000 : 20000)$
for $j = l$ to $l + Q$ **do**
  $d'_{i,j} = X$;
end for
(for NOISE faults:)
$m = Random(1 : n)$;
for $j = m$ to $m + R$ **do**
  $f = Random(.05 : 10)$;
  $d'_{i,j} = d_{i,j} + d_{i,j} \times f$;
end for

Fault Model. Since injecting faults in real implementations is difficult, it was decided to insert faults artificially according to the methodology described in Sharma et al. [2010].

The operational phase was modeled by dividing the entire dataset into small subsets. The starting point of each subset was selected randomly. Each subset contained 5,000 consecutive readings. Considering the enormous size of the collected dataset, 200 subsets were selected randomly to test the accuracy of PLA. Three types of fault were modeled, as described next and in Algorithm 2.

—Short. For each short fault, a random number k was generated to select a particular data point $d_{i,k}$. The value of $d_{i,k}$ was replaced by $d_{i,k} + d_{i,k} \times f$, where f is a random multiplicative factor in the range 0.1 to 10. In every iteration, both the data point and multiplicative factor were selected randomly.
—Constant. For constant faults, the starting data point $d_{i,l}$ was selected randomly, similar to short faults. Data point replacement was carried out in the same way

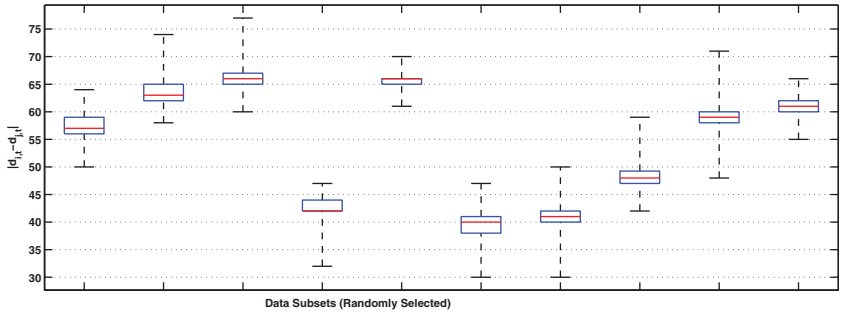

Fig. 10. Summary of relative difference between source and verifier data.

except that the affected data points were replaced with a large constant factor X. The value of X was much larger than its natural range. Q consecutive readings were replaced with constant faults, where the value Q was supplied by the user.

—Noise. Noise faults were introduced in the same manner as short faults except R consecutive readings were replaced using different multiplicative factors f at each iteration.

It is important to mention that drift faults were not inserted separately, rather noise faults (line 11–14) with a multiplicative factor f close to .05 exhibit the properties of a typical drift fault.

Metrics. The performance of PLA was assessed by determining the percentage of data points correctly attested faulty t_p (true positives), the percentage correctly attested as non-faulty t_n (true negatives), the percentage incorrectly attested as faulty f_p (false positives), and the percentage incorrectly attested as non-faulty f_n (false negatives). Overall, the accuracy a of PLA was defined as follows [Han 2005].

$$a = \frac{t_p}{(t_p + f_p)}.$$

Simulations. Two scenarios were considered: faults at the source only and faults at both the source and verifier.

Case I.. In the case of source-only faults, all three fault models were used. For each fault type, 500 faults were injected randomly into every subset. Thus, each sequence contains 10% faults.

Case II.. In the case of faults at the source and verifier nodes, 10%, faults were inserted in source node data subsets, as with Case I. Additionally, 2% short faults were inserted into each verifier subset. No noise or constant faults were inserted since in PLA these faults are checked in the verifier node, as described in Section 4.2. As soon as these fault types are experienced, the source is informed and node switching is performed to avoid data attestation by a faulty node.

Results. In the case of source-only faults, PLA provided 100% accuracy for fault detection for all 200 data subsets. The PLA data verification process depends on both corresponding verifier data ($d_{j,t}$) and the MaxDiff value $Z_{i,j}$. The results of $|d_{i,t} - d_{j,t}|$ for each of the 200 subsets were examined and found to be within the limits of MaxDiff value $Z_{i,j}$ (i.e., 107). Figure 10 shows the $|d_{i,t} - d_{j,t}|$ in the form of a box plot when both source and verifier data are non-faulty. Since similar results were obtained for all 200 subsets, the figure only shows ten subsets chosen randomly. It assures that faults at the source node will be detected provided that (i) the Verifier data is non-faulty,

Table VI. Statistics for False Positive and False Negative

	False Positive		False Negative	
	No.	In %	No.	In %
Maximum Value	50	10.00	19	3.80
Minimum Value	8	1.60	2	00.40
Mean Value	26	5.20	9	1.80

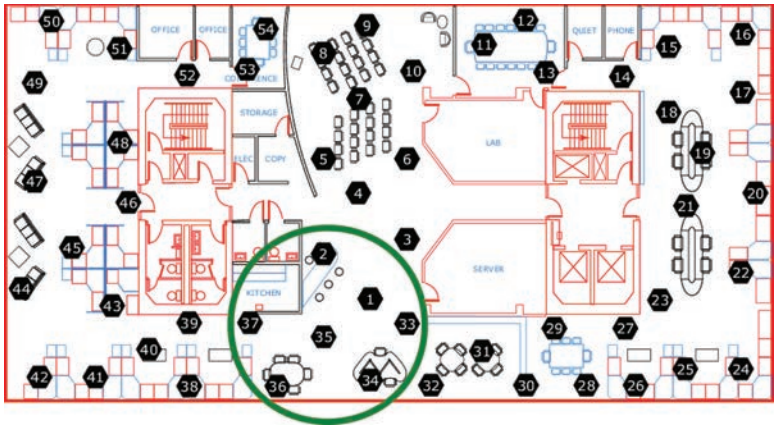

Fig. 11. Intel lab layout.

(ii) the MaxDiff value is computed correctly, and (iii) the magnitude of the fault is distinguishable from its expected value. As a result, after fault insertion in the source data, we found $|d'_{i,t} - d_{j,t}| > Z_{i,j}$ for all 500 faulty data points in every subset.

The results obtained for the source and verifier faults are listed in Table VI. False positives were found when verifier data was faulty but the source data was not. PLA experienced minimum, maximum, and average false positives as 8, 50, and 26, respectively. This decreased the detection precision very insignificantly (i.e., 00.52% on average). PLA also incurred an average of nine false negatives for each subset when both data points were faulty simultaneously and had similar direction. As a result, the MaxDiff value $Z_{i,j}$ failed to detect faults, since the relative difference of the two faulty data points were still within its threshold limit. Thus, the faults were missed.

In fact, the probability of such occurrence is very low since PLA works on the basis of 1-1 correspondence and faults are not generally spatially correlated. Moreover, the framework has an effective mechanism to detect noise and constant faults at the VN. A VN can not verify data if it is reported to be faulty.

5.2.2. Efficiency of PVN Selection. The efficiency of PVN selection was measured using the data gathered from the sensor network deployed in the Intel Berkeley Research lab between February 28, and April 5, 2004 [Intel Data 2004]. The sampling period was 31 seconds. For the experiment, Node 35 was selected as the source, while its surrounding nodes (i.e., 2, 1, 33, 34, 36, 37) were used as candidates for PVN. The layout of the network is given in Figure 11, where the green circle indicates the area used for the evaluation.

The objective of this section is to model the scenario in which Nodes 1 and 33 are faulty (in LP) and the others are non-faulty. For this scenario, we compared the performance of PVN selection with a random VN selection process. Once a PVN was selected (either by PVN or random selection), we carried out the same procedure to insert faults at both the source and verifier, as described in Section 5.2.1. Finally, the fault detection

Table VII. Data Statistics for LP

Category	Node ID	Mean	Variance
Non Faulty	2	6,394.12	4,037.07
	34	6,538.42	4,073.43
	35	6,508.64	4,071.78
	36	6,588.44	4,087.58
	37	6,513.87	4,113.07
Faulty	1	8,353.46	4,055.89
	33	9,477.24	9,507.54

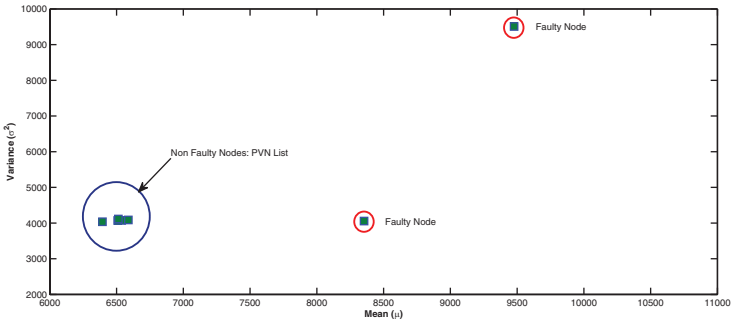

Fig. 12. Node clustering by PVN.

accuracy was measured. We repeated the entire process 50 times and averaged it as the final finding.

Data Preprocessing. We used Data Epoch No. 1,000 to 6,000 for each node data. The same epoch number at different nodes indicates data collected at the same time. The following three steps were performed for preparation of the dataset.

Step 1. Nodes 1 and 33 were found faulty in 13% and 26%, namely before any alteration of the data. We repositioned the faults for these two nodes so that their initial data points (used for LP) experienced faults at a similar rate (i.e., 13% , 26%).

Step 2. Missing values for all nodes were replaced by one of the following.

(a) Mean ($d_{i,t-1}$+$d_{i,t+1}$) if $d_{i,t}$ were missing, provided that both $d_{i,t-1}$ and $d_{i,t+1}$ were present and non-faulty.
(b) Median of each node's data if several consecutive values were missing.

Step 3. All the nodes experienced constant faults during the selected portion of the dataset. To ensure all nodes except Nodes 1 and 33 were non-faulty during LP, each constant fault was replaced with its respective median value.

As a result, we obtained our desired scenario where two nodes (i.e., 1, 33) were faulty, while all others (i.e., 2, 34, 36, 37) were not during the learning phase.

Duration of LP. The LP was performed using the first 500 samples from each node. The LP identified a list of the PVN based on the data statistics collected from this training data. This is given in Table VII. Here, each data was converted to its original ADC value to maintain coherence with the previous section. Clearly, Nodes 1 and 33 were identified as non-qualified members for the PVN list, since their data statistics were significantly different then those of other nodes. Graphically, the results are shown in Figure 12. Thus, Nodes 2, 34, 36, 37 constituted the PVN list.

Impact on Fault Detection. In order to assess the benefit of the PVN algorithm, we conducted the fault detection process for 50 rounds. In each round, faults were inserted at both the selected VN and source node following similar steps to these of the verification accuracy section. Data points between epoch 1,500 to 6,500 were used for this purpose. When a faulty node was selected by random selection, the accuracy dropped significantly. For instance, Node 1st detection accuracy ranged from 75.00% to 89.75%, while Node 33 achieved an accuracy of 41.60% and 71.40% in its worst and best cases, respectively. For all other nodes, the detection accuracy was in the range of 97.40% to 99.20%, which was almost identical to the result of the previous section.

Finally we averaged the accuracy in both methods. When a node was selected by the PVN algorithm, the average fault detection accuracy was 98.30%. But if a node was picked randomly, the average accuracy reduced to 88.13%. Clearly, the PVN algorithm has a direct effect on the fault detection accuracy.

5.3. Number of Bytes Transmitted

PLA is an in-network framework for verification of each sensed data by its peer node. Consequently it avoids redundant transmissions, which prolongs network lifetime. This section provides numerical calculations to measure the benefit of PLA in terms of the total number of bytes transmitted for a particular data request.

Notation. The following network parameters are used in the following analysis.

R = total number of sensor readings requested, called *Request Size*.
n = the number of readings in a single packet.
s = size of the data payload of a packet (in bytes).
c = size of the packet overhead (in bytes).
H = the average number of 1-hop neighbors of a Source Node

In this comparison, both link layer and routing layer retransmission and acknowledgement are ignored, since they are identical for all three methods. The comparison was done with two conventional methods which are defined as follows.

Method A. Most of the works [Mukhopadhyay et al. 2009; Sharma et al. 2010; Tulone and Srivastava 2007] on sensor data reliability are done with an assumption that data from nearby sensors is available. This implies that both the source and verifier send data to the Basestation at the same time. Therefore, the total number of bytes transmitted for request size R is given by

$$S_{methA} = 2(R/n)(c + s). \tag{5}$$

Method B. Guestrin et al. [2004] and Kamal et al. [2010], minimize the transmission of data packets by only sending the coefficients of a linear approximation to the data. Linear approximation is performed at the verifier, since the source data is intended for use in the application and may not tolerate approximation.

So

$$S_{methB} = (R/n)(c + s) + \lambda \times (R/n)(c + s), \tag{6}$$

where λ is the approximation ratio.

PLA. During the learning phase, three messages are used: (i) statistical parameters to the SN, (ii) request messages to the VN, and (iii) reply messages from the VN. These messages are exchanged with all one-hop neighbors. During the operational phase, each reading is only sent to the verifier node for checking.

Therefore, the total bytes transmitted S_{pla} for a particular request size R is

$$S_{pla} = 3 \times H \times (c + s) + (R/n)(c + s). \tag{7}$$

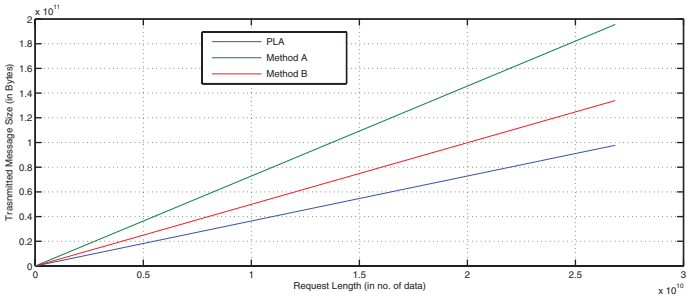

Fig. 13. Reduced message transmission.

Note that in both Methods A and B there is no additional local transmission, as they both rely on sense-and-send structures. PLA, on the other hand, counts both local transmissions during LP and actual data transmission during OP.

Comparison. Figure 13 shows how the number of bytes transmitted varies with the increasing value of the request size R for the three methods. It is assumed that IEEE standard 802.15.4 is used for data transmission [IEEE 2006]. Hence, $c = 13$ based on the format of 802.15.4 packets. Based on the implementation, as described in Section 5.1, $s = 42$ for PLA and $s = 38$ for the other methods, because PLA incurs an additional four bytes for status bits and verifier address. We used $\lambda = 1.37$, as found in Kamal et al. [2010]. It is also assumed that a moderate network size is used, hence $H = 10$. For the request size exceeding 1,000 readings, PLA achieves savings of 41.25% and 14.25% when compared with Methods A and B, respectively. This represents a large savings in radio message transmission, significantly prolonging the network lifetime.

5.4. Network Scalability

The performance of PLA was assessed in simulation for large network sizes.

Avrora (Beta 1.7.105) [Avrora 2008] was used for simulation. Since Avrora uses identical program images as those described in the implementation section, it reduces the possibility of experimental errors in moving from a real-life deployment to a simulation environment. The simulator models the use of Micaz motes and CC1000 radios. Simulations were performed on a server running Linux 2.6.26. The server was equipped with two 3.0 GHz CPUs and 8 GB of RAM. Sun Java 1.6.0 was used to run the experiments.

Simulation Parameters. As explained in Section 3, a verifier node may have a one-to-one configuration or a many-to-one configuration depending on the location of the nodes in the network. Both configurations are considered in this context.

A 10×10 two-dimensional grid was used for positioning nodes. An external topology file was used to build the network structure. The placement of source nodes and their corresponding verifier nodes was controlled manually using a topology file to ensure that *m-1* and *1-1* configurations were maintained for each combination of nodes. The basestation (Node ID 1) was placed at the origin $(0, 0)$. Other nodes were given IDs between 2 and 100. Relay nodes were placed randomly. CTP was used as the multihop routing protocol. No duty cycling was performed at the MAC layer. A total of five experiments were conducted, and the average values are shown in the result section. A summary of the simulation parameters is presented in Table VIII.

The syntax $< s, v, r, b >$ is used to describe the network topology.

—s = the number of source nodes in the network.
—v = the number of verifier nodes in the network.

Table VIII. Summary of Simulation

Parameter or Attribute	Value
Total No. of Nodes instantiated	10 to 100
No. of BS	1 (with Node ID 1)
MAC Layer	Default CSMA with LPL disabled
Routing Layer	CTP
RF Transceiver	CC1000 AM
Radio Propagation	Free-space Radio Propagation Model
Data Payload Size	42 bytes
Duration of Simulation	10 Seconds
No. of Iteration	5 times

Table IX. Node Distribution for simulation

No. of Node N	1-1 Config. (s,v,r,b)	m-1 Config. (s,v,r,b)
10	2,2,5,1	2,1,6,1
20	4,4,11,1	4,1,14,1
30	6,6,17,1	6,1,22,1
40	8,8,23,1	8,1,30,1
50	10,10,29,1	10,2,37,1
60	12,12,35,1	12,2,45,1
70	14,14,41,1	14,2,53,1
80	16,16,47,1	16,2,61,1
90	18,18,53,1	18,3,68,1
100	20,20,59,1	20,3,76,1

—r = the number of relay nodes in the network.
—b = number of basestations in the network.

The total number of nodes in the network is $N = s + v + r + b$. The exact distribution of the nodes in both configurations is given in Table IX. Program images for source nodes, verifier nodes, relay nodes, and basestation, as described in Section 5.1, were used for simulation.

Results. For each value of N, every node ran for ten simulated seconds. An increasing value of N indicates higher workload on larger networks in terms of simulation cycles. Figure 14 shows the variation of simulation execution time with the number of nodes in the network. Unlike event-based simulators, Avrora captures time for sensing, internal computation, and communication. In both *1-1* and *m-1* configurations, execution time was linear with increasing network size.

The savings in *m-1* configurations come from the fact that when multiple sources send data packets to a single verifier node at the same time, the verifier can sense once and attest several received packets without any accuracy penalty. Thus, the verifier avoids repeated sensing.

Since Avrora is a cycle-accurate simulator the overall network performance was measured in megahertz (MHz), that is, 10^6 cycle/second (rather than MIPS). It considers the cost of both simulating device operation and sleeping, which MIPS measure can not do [Titzer et al. 2005]. If there are n nodes in the network and each node N_i executes C_i cycles during T_i seconds then

$$\text{Network Performance (in MHz.)} = \sum_{i=1}^{n} (C_i / T_i) \times 10^6.$$

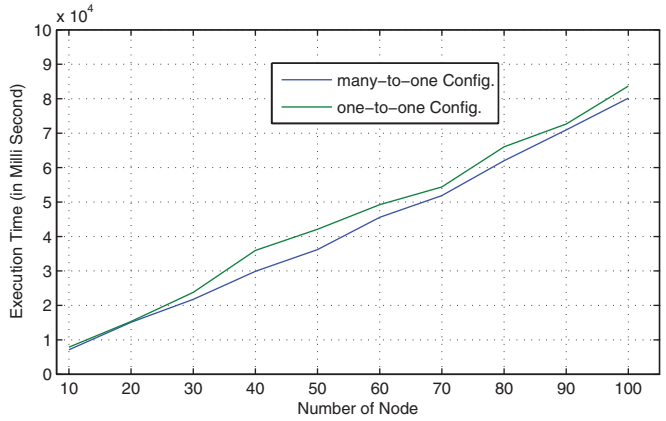

Fig. 14. PLA for a larger network.

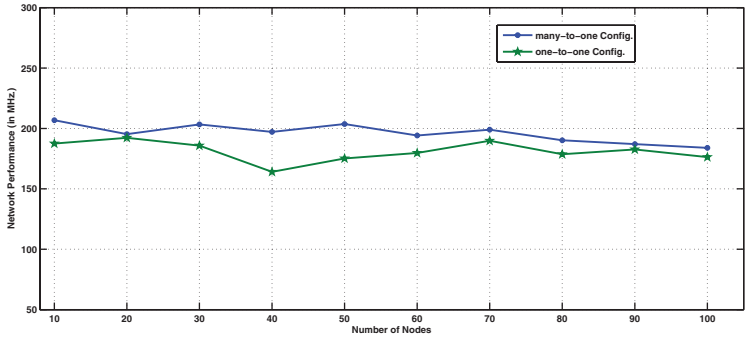

Fig. 15. Network performance stability.

Figure 15 explains the network performance. In both configurations, simulation results show stability in their performances. On average, the performances of both networks (i.e., *1-1* and *m-1*) were measured as 181.19 MHz and 196.10 MHz with increasing network size.

These findings indicate that the PLA framework can be deployed in a large network without significant penalty in terms of response time and network performance.

6. RELATED WORK

To the best of our knowledge, PLA is the first work that proposes certification of sensor data by exploiting spatial correlation at one-hop neighbor nodes. The framework has been shown to be accurate, lightweight, and scalable.

A number of research works have proposed server-side sensor faults detection techniques in WSNs. Previously, both auto regressive integrated moving averages (ARIMA) and hidden Markov models (HMM) have been combined with rule-based techniques, such as histogram-based methods [Ramanathan et al. 2006]. These hybrid methods [Sharma et al. 2010] show increased fault detection ratios. However, the HMM-based models require large datasets for training. Missing and faulty data detection in WSN has been studied in Koushanfar and Potkonjak [2005]. Their paper describes use of a Markov chain model in identifying faulty nodes. Data analysis was done on the server side after a data collection phase. Server-side implementation is required since formation of the n-step transition matrix from the initial transition matrix is beyond the

capability of microcontroller-based sensor nodes. A solution, as proposed in Elnahrawy and Nath [2003], assumes that the knowledge of sensor data and error distribution are known a priori, which may not be the case in real deployments. A calibration method such as that of Bychkovskiy et al. [2003] addresses permanent faulty measurements using two-phase post-deployment technique. The first phase of the technique exploits temporal correlation at nearby sensors. The second phase is formulated as an optimization problem and a distributed solution is presented. The major limitation of this work is that it can handle only calibration error with limited magnitude. Faulty sensor readings and node failures also affect the accuracy of event detection. It has been investigated [Krishnamachari and Iyengar 2004] in which a distributed bayesian algorithm for the purpose of efficient event detection is intilized.

The method proposed in Balzano and Nowak [2007] exploits the linearity of sensor data and models a network with n sensors. The solution is theoretically sound but lacks practical implementation. Furthermore, it recommends that sensor data should be present at the server side from all possible sources at the same time, which shortens the network lifetime, especially in the case of heavy traffic in a large network. In general, the disadvantage of server-side solutions is that they require transmission of large quantities of data to the server, which increases network congestion and reduces network lifetime. Server-side data analysis algorithms are typically not useful in the distributed case since they are too computational complex and/or require too much memory.

Distributed sensor-data checking for WSNs was addressed in Tulone and Srivastava [2007]. Therein, the concept of a virtual reference sensor (VRS) was adopted. It uses a probabilistic forecasting method to build reference values. Although the method can classify faults, the construction of a VRS involves quorum-based bulk data reporting by several sensors, simultaneously to the sink, which degrades network life time. Another system to deal with faulty nodes in WSNs was reported in Guo et al. [2009]. The method focuses on checking acoustic signals. The authors exploit the assumption that acoustic sensor readings monotonically decrease, or weaken, as the distance between nodes increases. The system output is a probabilistic trust factor for each node, a fraction between 0 to 1, based on the Demster-Shafer belief theory [Commerce et al. 2002]. A reputation metric based on past activities is used for prediction of reliability. The work relies on checking using simultaneous readings, rather than predictions based on a historical record.

There have been many publications on reliability from the routing perspective (e.g., [Gnawali et al. 2009; Puccinelli and Haenggi 2010; Moeller et al. 2010]. The core purpose of all these works is to enhance the packet reception ratio in highly dynamic and lossy environments. Gnawali et al. [2009] proposed a multihop routing protocol, Collection Tree Protocol (CTP), which handles link dynamics and topology changes with minimum overhead. The protocol proposed in Puccinelli and Haenggi [2010] is founded upon the principle that routing over a few long hops is more reliable than routing over a much higher number of short hops. The most recent work in this area is the Backpressure Collection Protocol (BCP) [Moeller et al. 2010] which reduces end-to-end packet delays by using a last-in-fist-out queue instead of a conventional fist-in-fist-out queue.

Various data aggregation policies have been proposed to minimize the penalty incurred by faulty sensor readings. Fan and Chen [2010] propose tree-based aggregation with multipath data forwarding to handle sporadic communication failure. Additionally, they reduce the problem of duplicate data at the aggregation point by means of a linear counting technique. The same issue was addressed in Hellerstein et al. [2003], where the solution largely relies on SQL-like data aggregation. This can minimize the error in data aggregation in the face of some erroneous readings based on wavelet

histograms. A major limitation of these solutions lies in the fact that data aggregation may not be desirable in many applications of WSN.

A recent work related to our framework is Liu et al. [2010]. The authors explore the concept of packet marking, similar to setting the status bits of PLA. However, the work is aimed at passive diagnosis of network topology by means of assessing the routing paths by stamping the data packet. The concept of exploitation of spatial correlation in WSN data has been extensively studied in previous works [Vuran et al. 2004]. However, these previous works focus on other applications, such as load balancing by efficient scheduling of data collection [Lee et al. 2010].

7. CONCLUSIONS AND FUTURE WORK

Although there has been a substantial amount of research on assessing and enhancing the reliability of WSN routing, there has been comparatively little work on assessing and enhancing data reliability in WSNs. Most previous work in the area focuses on server-side solutions. In this article, we propose a distributed data reliability assessment framework named PLA. One-hop neighbor nodes are used to check and attest for the results of queries at source nodes. Attestation is achieved by piggybacking an attestation bit in the data packet transmitted by the source.

The accuracy, network transmission overhead, computational complexity, power consumption, and memory consumption were assessed. The performance of the framework was also compared with that of previous methods. The framework was implemented in a testbed network. It was found that PLA achieved over 99% fault detection precision at the cost of only 1.45% packet loss due to node-level processing. Further, PLA utilized the PVN selection that achieved over 10.00% improvement in fault detection compared with random selection. Overall, the method incurs up to 41% fewer message transmissions than previous methods.

In future work, we plan to investigate combinations of PLA with mechanisms for enhancement of routing reliability and data security. In medical applications [Chipara et al. 2010; Ko et al. 2010], for example, the data and attestation would have to be secured from any illegal disclosure, fabrication, and modification. A major challenge in this context lies in devising cryptographic primitives which can be efficiently implemented in conjunction with PLA in resource-constrained sensors.

REFERENCES

AVRORA 2008. The AVR simulation and analysis framework. http://compilers.cs.ucla.edu/avrora/. (Last. accessed 11/10).

BALZANO, L., AND NOWAK, R. 2007. Blind calibration of sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN'07)*. 79–88.

BRICKELL, E., CAMENISCH, J., AND CHEN, L. 2004. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04)*. 132–145.

BYCHKOVSKIY, V., MEGERIAN, S., ESTRIN, D., AND POTKONJAK, M. 2003. A collaborative approach to in-place sensor calibration. In *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks (IPSN'03)*. 301–316.

CHIPARA, O., LU, C., BAILEY, T. C., AND ROMAN, G.-C. 2010. Reliable clinical monitoring using wireless sensor networks: Experiences in a step-down hospital unit. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys'10)*. 155–168.

COMMERCE, B. E., JSANG, A., AND ISMAIL, R. 2002. The beta reputation system. In *Proceedings of the 15th Bled Electronic Commerce Conference*. 324–337.

CORKE, P., WARK, T., JURDAK, R., HU, W., VALENCIA, P., AND MOORE, D. 2010. Environmental wireless sensor networks. *Proc. IEEE 98,* 11, 1903–1917.

CROSSBOW 2010. Data sheet from Crossbow. http://www.xbow.com/Products/productdetails.aspx?sid=252. (Last accessed 3/10).

Deshpande, A., Guestrin, C., Madden, S. R., Hellerstein, J. M., and Hong, W. 2004. Model-driven data acquisition in sensor networks. In *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB'04)*. 588–599.

Elnahrawy, E. and Nath, B. 2003. Cleaning and querying noisy sensors. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA'03)*. 78–87.

Fan, Y.-C. and Chen, A. L. P. 2010. Efficient and robust schemes for sensor data aggregation based on linear counting. *IEEE Trans. Parallel Distrib. Syst. 21,* 11, 1675–1691.

Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., and Levis, P. 2009. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*. 1–14.

Guestrin, C., Bodik, P., Thibaux, R., Paskin, M., and Madden, S. 2004. Distributed regression: An efficient framework for modeling sensor network data. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*. 1–10.

Guo, S., Zhong, Z., and He, T. 2009. Find: Faulty node detection for wireless sensor networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*. 253–266.

Han, J. 2005. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Hellerstein, J. M., Hong, W., Madden, S., and Stanek, K. 2003. Beyond average: Toward sophisticated sensing with queries. In *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks (IPSN'03)*. 63–79.

IEEE. 2006. IEEE Standard 802.15.4. http://www.ieee.org/index.html.

Ingelrest, F., Barrenetxea, G., Schaefer, G., Vetterli, M., Couach, O., and Parlange, M. 2010. Sensorscope: Application-specific sensor network for environmental monitoring. *ACM Trans. Sen. Netw. 6*, 17, 1–32.

Intel. 2004. Intel lab sensor data. http://db.csail.mit.edu/labdata/labdata.html.

Kamal, A. R. M., Razzaque, M. A. A., and Nixon, P. 2010. 2pda: Two-phase data approximation in wireless sensor network. In *Proceedings of the 7th ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN'10)*. 1–8.

Ko, J., Lim, J. H., Chen, Y., Musvaloiu-E, R., Terzis, A., Masson, G. M., Gao, T., Destler, W., Selavo, L., and Dutton, R. P. 2010. Medisn: Medical emergency detection in sensor networks. *ACM Trans. Embed. Comput. Syst. 10*, 11, 1–29.

Koushanfar, F. and Potkonjak, M. 2005. Markov chain-based models for missing and faulty data in mica2 sensor motes. In *Proceedings of the IEEE Sensors Conference*.

Krishnamachari, B. and Iyengar, S. 2004. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Trans. Comput. 53*, 241–250.

Lee, H., Keshavarzian, A., and Aghajan, H. 2010. Near-lifetime-optimal data collection in wireless sensor networks via spatio-temporal load balancing. *ACM Trans. Sen. Netw. 6*, 26, 1–32.

Lim, J. C. and Bleakley, C. 2010. Robust data collection and lifetime improvement in wireless sensor networks through data imputation. In *Proceedings of the 5th International Conference on Systems and Networks Communications (ICSNC)*. 64–69.

Liu, C., Wu, K., and Pei, J. 2007. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Trans. Parallel Distrib. Syst. 18*, 1010–1023.

Liu, Y., Liu, K., and Li, M. 2010. Passive diagnosis for wireless sensor networks. *IEEE/ACM Trans. Netw. 18,* 4, 1132–1144.

Marti, S., Giuli, T. J., Lai, K., and Baker, M. 2000. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00)*. 255–265.

Moeller, S., Sridharan, A., Krishnamachari, B., and Gnawali, O. 2010. Routing without routes: The backpressure collection protocol. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'10)*. 279–290.

Mukhopadhyay, S., Schurgers, C., Panigrahi, D., and Dey, S. 2009. Model-based techniques for data reliability in wireless sensor networks. *IEEE Trans. Mobile Comput. 8,* 4, 528–543.

Ni, K., Ramanathan, N., Chehade, M. N. H., Balzano, L., Nair, S., Zahedi, S., Kohler, E., Pottie, G., Hansen, M., and Srivastava, M. 2009. Sensor network data fault types. *ACM Trans. Sen. Netw. 5*, 25, 1–29.

Puccinelli, D. and Haenggi, M. 2010. Reliable data delivery in large-scale low-power sensor networks. *ACM Trans. Sen. Netw. 6*, 28, 1–41.

Ramanathan, N., Balzano, L., Burt, M., Estrin, D., Harmon, T., Harvey, C., Jay, J., Kohler, E., Rothenberg, S., and Srivastava, M. 2006. Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks. *CENS Tech. Rep. 62,* .

SESNORSCOPE 2008. EPFL SensorScope Project. http://sensorscope.epfl.ch/index.php/Environmental_Data. (Last accessed 11/10).

SHARMA, A. B., GOLUBCHIK, L., AND GOVINDAN, R. 2010. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Trans. Sen. Netw. 6*, 23, 1–39.

SRINIVASAN, K. AND LEVIS, P. 2006. RSSI is under appreciated. In *Proceedings of the 3rd Workshop on Embedded Networked Sensors (EmNets'06)*.

STADDON, J., BALFANZ, D., AND DURFEE, G. 2002. Efficient tracing of failed nodes in sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*. 122–130.

SZEWCZYK, R., MAINWARING, A., POLASTRE, J., ANDERSON, J., AND CULLER, D. 2004. An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*. 214–226.

TAVAKOLI, A., KANSAL, A., AND NATH, S. 2010. On-line sensing task optimization for shared sensors. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'10)*. 47–57.

TINYOS 2010. TinyOS documentation. http://docs.tinyos.net/index.php/Main_Page. (Last accessed 1/10).

TITZER, B. L., LEE, D. K., AND PALSBERG, J. 2005. Avrora: Scalable sensor network simulation with precise timing. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*. 477–482.

TOLLE, G., POLASTRE, J., SZEWCZYK, R., CULLER, D., TURNER, N., TU, K., BURGESS, S., DAWSON, T., BUONADONNA, P., GAY, D., AND HONG, W. 2005. A macroscope in the redwoods. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys'05)*. 51–63.

TULONE, D. AND SRIVASTAVA, M. 2007. Inspect: A general framework for on-line detection and diagnosis of sensor faults. In *Proceedings of the 2nd International Conference on Internet Technologies & Applications (ITA)*.

VURAN, M. C., AKAN, Ö. B., AND AKYILDIZ, I. F. 2004. Spatio-temporal correlation: Theory and applications for wireless sensor networks. *Comput. Netw. 45*, 245–259.

XU, N., RANGWALA, S., CHINTALAPUDI, K. K., GANESAN, D., BROAD, A., GOVINDAN, R., AND ESTRIN, D. 2004. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*. 13–24.

ZHAO, F. AND GUIBAS, L. J. 2004. *Wireless Sensor Networks: An Information Processing Approach* 1st Ed. Elsivier, NY, Chapter 1, 4–10.

Received April 2011; revised November 2011; accepted November 2011