Energy saving models for wireless sensor networks

Energy saving models for wireless sensor networks

Daniele Apiletti, Elena Baralis, and Tania Cerquitelli.

Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Italy
Email: {daniele.apiletti,elena.baralis,tania.cerquitelli}@polito.it

Keywords Data Mining, Wireless sensor networks, clustering algorithm, energy saving models

Abstract *Nowadays wireless sensor networks are being used for a fast-growing number of different application fields (e.g., habitat monitoring, highway traffic monitoring, remote surveillance). Monitoring (i.e., querying) the sensor network entails the frequent acquisition of measurements from all sensors. Since sensor data acquisition and communication are the main sources of power consumption and sensors are battery-powered, an important issue in this context is energy saving during data collection. Hence, the challenge is to extend sensor lifetime by reducing communication cost and computation energy.*

This paper thoroughly describes the complete design, implementation and validation of the SeReNe *framework. Given historical sensor readings,* SeReNe *discovers energy saving models to efficiently acquire sensor network data.* SeReNe *exploits different clustering algorithms to discover spatial and temporal correlations which allow the identification of sets of correlated sensors and sensor data streams. Given clusters of correlated sensors, a subset of representative sensors is selected. Rather than directly querying all network nodes, only the representative sensors are queried by reducing the communication, computation and power costs. Experiments performed on both a real sensor network deployed at the Politecnico di Torino labs and a publicly available dataset from Intel Berkeley Research lab demonstrate the adaptability and the effectiveness of the* SeReNe *framework in providing energy saving sensor network models.*

1 Introduction

A sensor network acquires measurements of real world phenomena at discrete points. Each measurement is performed by a sensor and is characterized by a specific time and location of acquisition. Sensing technologies have developed new smart wireless devices characterized by computational, communication, and sensing capabilities. These devices, exploited in rather diverse applications (e.g., habitat monitoring Szewczyk et al (2004), highway traffic monitoring Gehrke and Madden (2004), remote surveillance He et al (2004)) continuously monitor a given environment. To effectively accomplish this task, the sensor network is frequently queried, i.e., acquisition from all sensors of measurements describing the state of the monitored environment Gehrke and Madden (2004); Madden et al (2003); Yao and Gehrke (January 2003) is performed. However, this approach is characterized by high energy consumption. Since main contributors to sensor energy consumption are communication and data acquisition Deshpande et al (2004), novel intelligent techniques for sensor network querying are needed. Hence, this work has been focused on devising power-efficient models for energy saving during data collection.

This paper thoroughly describes the SeReNe framework which provides energy saving models for sensor networks. The models can be exploited to efficiently acquire sensor data by minimizing communication cost. SeReNe exploits clustering techniques to study temporal correlation among sensor data streams and spatial correlation among sensor nodes. Given clusters of correlated sensor data, the "best" subset of nodes, possibly including outliers and representing all sensors, is singled out. Rather than directly querying all network nodes, only the representative sensors are queried to reduce the communication and power costs. Furthermore, since a query optimizer aims at identifying the cheapest execution plan according to an estimated cost, SeReNe is profitably exploited also in this context. Given a set of representative sensors identified by SeReNe, a schedule minimizing the acquisition cost may be computed, for instance, by means of a TSP solver Skiena (2008). Experiments performed on both a real sensor network deployed at the Politecnico di Torino labs and a publicly available dataset from Intel Berkeley Research lab Intel (2009) show the effectiveness and efficiency of the SeReNe framework in characterizing energy-aware models for sensor network data.

The paper is organized as follows. Section 2 presents an overview of the SeReNe framework and describes the main features of its building blocks, whose implementation is discussed in Section 3. Section 4 presents and analyzes experiments performed to validate the proposed framework. Section 5 discusses related work and compares our approach with previous ones, while Section 6 draws conclusions and discusses future work.

2 The SeReNe framework

SeReNe (SElecting REpresentatives in a sensor NEtwork) is an environment for the identification of energy saving models to efficiently query sensor networks. Figure 1 shows the SeReNe framework integrated into a sensor network architecture. Sensor nodes frequently receive queries from the sink (i.e., base station). Each node performs the query over its sensor data and the result is sent to the base station by means of a multi-hop data collection protocol Madden et al (2003). Before transmitting the query results, sensor data can be sometimes compressed by means of reduction/compression techniques Deligiannakis et al (2004); Tang and Raghavendra (2004). These in-network processing techniques are a complementary approach that address the sensor network efficiency at a different level and that can be successfully applied besides SeReNe.

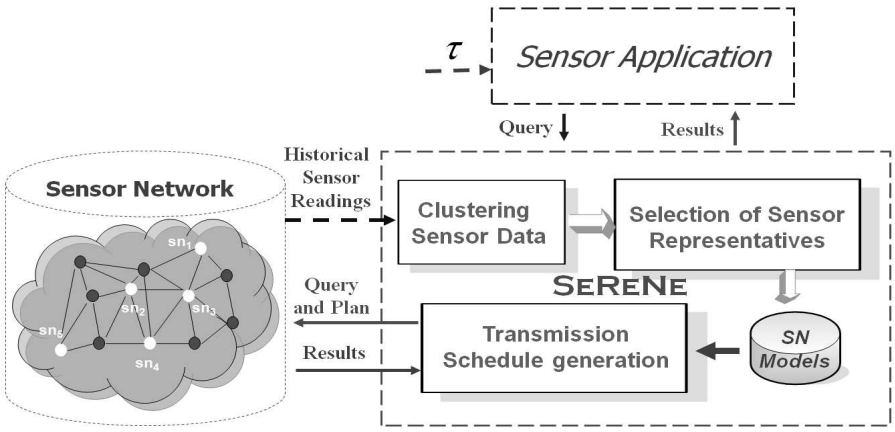


Figure 1: Architecture of the SeReNe framework

SeReNe generates sensor network models by means of two steps: (1) Correlation analysis, and (2) Selection of sensor representatives. The correlation analysis block discovers temporal correlations among sensor data

streams, in terms of correlated sensors, correlation time and strength. Furthermore, it discovers spatial correlations among faraway and neighboring sensors. Given a set of correlated sensors, the second phase allows singling out a subset of sensors from the network, called Representative-Sensors (R-Sensors), which best represent all network nodes. During the selection of R-Sensors, several criteria may be considered (e.g., distance and transmission cost among sensors). The time window in which the sensor network model is effectively representing the network state is computed by guaranteeing a user-provided error bound τ (see Section 2.3 for further discussion). Furthermore, the network model can be reliably adapted to network changes by continuously analyzing data collected through the network itself (see Baralis et al (2007) for a further discussion), thus allowing the detection of deviations from the modeled behavior.

A sensor network model can be appropriate to execute some queries efficiently and accurately. Queries, exploited to monitor a sensor network, usually compute aggregation functions (e.g., MIN, MAX, AVG) for each measure. We use the AVG as the main aggregate operator to describe, analyze, and evaluate the SeReNe framework. Thus, the AVG operator is implied in all discussions unless other operations are explicitly stated.

Sensor network models generated by SeReNe are exploited to efficiently query the network. Smart sensors are characterized by computational, communication and sensing capabilities, thus allowing in-network data processing. However, exploiting SeReNe models, each time a query is executed, only R-Sensors are queried, since they represent the network state for the corresponding query. To collect sensor data, the query and the execution plan are broadcast to (interested) sensors. Hence, a transmission schedule needs to be generated. The transmission schedule generation block of the SeReNe framework (see Figure 1) aims at identifying an energy saving schedule which minimizes communication costs. Different optimization algorithms Con (2011); Skiena (2008); Helsgaun (2000) have been integrated to identify an energy saving execution plan (see Section 2.4 for further discussion).

2.1 Clustering sensor data

Correlation analysis on sensor measures detects relationships, both in the space and time dimensions, among physical phenomena and sensor data. SeReNe exploits clustering algorithms to discover correlations among sensor data streams.

Clustering algorithms group objects into sets (clusters) on the basis of information related to objects. The aim is to obtain clusters of similar objects, whereas objects in different clusters should be dissimilar. The larger the similarity among objects within a cluster, and the larger the difference between clusters, the higher the clustering quality. Similarity is normally measured according to a notion of distance among objects and clusters. Many techniques have been proposed in literature to cluster data. General features which characterize each approach are (i) the amount of knowledge needed to correctly set the input parameters, (ii) the ability to deal with clusters of different shapes, (iii) the ability to deal with noisy data, i.e. the sensitivity to outliers, missing or erroneous data.

Clustering algorithms, exploited to analyze sensor data, lead to a better identification of correlated groups even when sensors are physically far away and/or sensor data are collected in distant time instants (e.g., measurements collected in different days). Two different analyses are addressed:

- *Physical correlation analysis* allows discovering the similarity of the environment where the sensors are located. Two different cases can be discovered: (i) Two sensors located nearby sense similar values (e.g., sensors 1 in room A and 2 in room B, both at the second floor, sense the same value of temperature from 1 pm to 3 pm since they are in direct sun light). (ii) Far sensors, located in similar environments, sense correlated measurements (e.g., sensors 3 in room C and 4 in room D sense the same value of temperature from 11:30 am until 1:30 pm since both rooms are crowded for lunch).

- *Time correlation analysis.* Sensor data streams may be correlated *over time*. By means of this type of analysis, we can discover: (i) Correlated phenomena (i.e., two phenomena follow a similar pattern evolution) and (ii) correlated measurements of the same environmental parameter (e.g., the variation pattern of the measurement, for example, every hour). By means of the former relationship, we can query the sensor network for only one phenomenon, whereas the latter can be useful to decrease query frequency.

We have analyzed three broad classes of clustering algorithms: (i) Partitioning approaches, (ii) density-based methods, and (iii) model-based methods. Partitioning-based and density-based methods require the definition of a metric to compute the distance between objects in the dataset. Partitioning-based methods are able to identify only spherical-shaped clusters (e.g., identify a center and a radius), unless the clusters are well separated, and are sensitive to the presence of outliers. The K-Means Juang and Rabiner (1990) approach is a popular method which belongs to this category. Density-based methods are designed to deal with non-spherical shaped clusters and to be less sensitive to the presence of outliers. The objective of these methods is to identify portions of the data measurement space characterized by a high density of objects. Density is defined as the number of objects which are in a given area of the measurement space. The general strategy is to explore the space by

growing existing clusters as long as the number of objects in their neighborhood is above a given threshold. The DBSCAN Ester et al (1996) algorithm is a classical representative of the density-based category. Finally, model-based methods assume that each cluster can be modeled by means of a mathematical model. Data are grouped in clusters to determine the best fit between mathematical model and the data. These algorithms are able to correctly take into account outliers and noise by making use of standard statistical techniques. Expectation-maximization (EM) algorithm G. McLachlan and T. Krishnan (1997) has been considered in this paper as a representative of the model-based category. EM algorithm performs statistical modeling assuming a Gaussian mixture object distribution. The EM algorithm computes the parameters of a parametric mixture model distribution (i.e., a probability density function which is expressed as a convex combination of Gaussian functions with weights).

Among the clustering algorithms, K-Means Juang and Rabiner (1990), DBSCAN Ester et al (1996), EM G. McLachlan an (1997), and many others available in the Weka Wek (2009) data mining library have been integrated in SeReNe to perform the correlation analysis among sensor data streams. Furthermore, in SeReNe, the distance between sensor data is measured by means of the Euclidean distance computed on normalized data. The effectiveness of clustering techniques in detecting sensor correlation is experimentally validated and discussed in Section 4.1.2 and Section 4.2.2.

2.2 Selection of sensor representatives

This step consists in singling out a subset of representative sensors, denoted as R-Sensors, from a set of sensor clusters. The subset may contain one or more sensors for each cluster according to the required model accuracy (i.e., the error bound τ). The number n of R-Sensors is set according to the required model accuracy. The number of representatives in each cluster is proportional to the number of cluster points. Reliable outliers are included in the R-Sensors.

For the modeled network behavior (see also Section 2.3 for discussion on the temporal validity of the model), the framework allows to limit the querying to the R-Sensors only, by guaranteeing the error bound τ. However, unseen deviations may occur on new data, e.g., a failure in a non-representative sensor. Different solutions can be applied to avoid losing these deviations, from sensors reporting to their cluster representative, to full data post processing for adapting the model to the new behavior.

Given a cluster of sensors, each of which senses k measures, the SeReNe framework exploits three selection strategies Baralis et al (2007) to single out the subset of sensors that better model the correlated group.

- *Strategy 1*, also called *Measure trend*, is based on the analysis of correlated phenomena. Both the physical clustering in a given sampling time and the measurements collected during the considered time period are considered to represent physical and temporal correlations among sensors and their data. The best approximation of phenomenon j over the time period is given by the average value of measurements collected by all sensors during the considered period, denoted as $\overline{M_j}$. Let $\overline{O} = (\overline{M_1}, \ldots, \overline{M_k})$, where k is the number of considered measures. Each sensor i is described by means of an array $M(i)$=$(\overline{M_{1i}}, \ldots, \overline{M_{ki}})$ computed over the time period. Representative sensors are the n nodes nearest to $\overline{O}$ that correspond to the minimum communication cost.

- *Strategy 2* and *Strategy 3*, also called *Cluster shape* and *Cluster shape and core* respectively, are based on the physical location of correlated sensors. These selection techniques focus on cluster shapes of correlated sensors. Cluster border nodes are exploited to select representatives, thus detecting cluster shapes. At first, we compute the cluster barycenter $(\overline{x}, \overline{y}, \overline{z}) = (\frac{1}{m}\sum_{i=0}^{m} x_i, \frac{1}{m}\sum_{i=0}^{m} y_i, \frac{1}{m}\sum_{i=0}^{m} z_i)$ where (x_i, y_i, z_i) are the spatial coordinates of sensor i in the considered cluster of m sensors. Next, sensor coordinates are normalized with respect to the barycenter by computing $(x_i - \overline{x}, y_i - \overline{y}, z_i - \overline{z})$ for each sensor s_i. In this way, the barycenter is regarded as as the reference system center. Distances between each normalized sensor and the barycenter are sorted in ascending order. A sensor is selected as representative if either its distance from all previously selected representatives is larger than its distance from the barycenter, or it is in a different quadrant with respect to all other representative sensors.

  Strategy 3 extends strategy 2 by including the closest-to-barycenter nodes as R-Sensors.

The effectiveness of the selection strategies in identifying a proper subset of sensors to represent the network state has been experimentally validated and discussed in Section 4.1.3 and Section 4.2.3.

2.3 Sensor network model validation and characterization

Different clustering sessions are performed to suitably characterize sensor correlations. Each session considers a different set of measure combinations (e.g., temperature, humidity), and performs two analyses : (i) Correlation over time and (ii) Physical correlation. In both cases, clustering techniques are exploited. During the first phase, time series collected by each sensor are clustered. A set of correlated sensor values for each sensor are

returned by the clustering algorithm. Clustering results are plotted on a mono-dimensional diagram where the x-axis represents time. On this graph, cyclically repeated time bands can be detected. Since the network consists of many sensors, clustering results obtained from each time series need to be merged, thus overlapping time bands are grouped together. For each group, the largest time band is considered to set the suitable validity window T_{model}. A sensor network model has to be built for each time band, leading to different physical correlations for each band. A physical correlation clustering session is performed separately for each time unit in the time band, and analyzes the values observed by all sensors, resulting in a set of sensor clusters. Thus, correlated measurements collected in the same time period by different sensors are clustered into the same group, independently of the spatial position of the sensing device. Each cluster set is evaluated by computing the overall cluster validity Pang-Ning Tan and Michael Steinbach and Vipin Kumar (2006) by means of a cohesion function, which is computed as the sum of the cohesion of individual clusters. The cluster set that maximizes the overall cohesion function will be exploited for building the sensor network model in the corresponding time band and for the selection of representative sensors.

After selecting the R-Sensors of a network state by means of one of the proposed strategies, the model validity window T_{model}, which is the temporal interval in which R-Sensors provide a good approximation of the network state (i.e., the error bound of the result is τ), needs to be computed. This is the largest subset of contiguous sampling times satisfying the threshold, given a subset of representative sensors. This time band, denoted as T_{model}, is the model validity window and is computed as follows. At first, we estimate the approximate value $\overline{Mr_j}$ of a measure j as the average on values collected by all representative sensors. The best approximation $\overline{M_j}$ is the average on the values gathered by querying all sensors. For each subset of contiguous sampling times we count the percentage of contiguous samples in which $\mid \overline{Mr_j} - \overline{M_j} \mid \leq \tau$. T_{model} is the largest subset of contiguous sampling times.

2.4 Transmission Schedule Generation

The transmission schedule generation block of the SeReNe framework (see Figure 1) generates an appropriate schedule among R-Sensors to minimize communication costs and balance energy consumption among sensors. The transmission schedule is a list of sensor nodes to be visited to collect sensor data. It is the cheapest round-trip route that visits each node exactly once and then returns to the starting node (i.e., base station, the node that interfaces the query processor to the sensor network).

Since we focus on networks with known topologies and unreliable communication based on acknowledgment messages and retransmission, we can model the network by means of a graph composed by a set of nodes (i.e., sensors) and a set of edges Deshpande et al (2004). Each edge between two nodes is characterized by a weight, which represents the average number of transmissions required to successfully complete the delivery. By considering p_{ij} and p_{ji} as the probabilities that a packet from sensor i will reach sensor j and vice versa, and by assuming that these probabilities are independent, the expected number of transmission and acknowledgment messages required to guarantee a successfully transmission between i and j is $\frac{1}{p_{ij} \cdot p_{ji}}$. This value (i.e., the edge weight) can be exploited to estimate the transmission cost required to exchange data between i and j.

Given the sensor graph, the transmission schedule that minimizes acquisition costs can be easily computed by means of a Traveling Salesman Problem (also called TSP) solver. Since solving the traveling salesman problem is NP-complete, many heuristics have been proposed in literature Skiena (2008) that are known to perform well in practice. By means of one of a TSP solver algorithm we can select the schedule minimizing the communication cost and balancing energy consumption among sensor. In particular, we integrated into SeReNe different algorithms to select an appropriate transmission schedule: (i) the Concorde algorithm Con (2011), an exact TSP solver for symmetric TSPs based on a branch-and-cut approach; (ii) the Heuristic Concorde algorithm Skiena (2008), an efficient and fast TSP solver using a branch-and-cut approach and the Chained Lin-Kernighan; (iii) the Genius algorithm Skiena (2008), based on TSP heuristic and designed for the vehicle routing problem; (iv) the LKH algorithm Helsgaun (2000), an efficient TSP solver using a branch-and-cut approach and Helsgaun's-Lin-Kernighan-variant. These approaches have been experimentally evaluated in Section 4.2.4.

3 SeReNe implementation

The SeReNe prototype has been designed to be fast, stable, easy to access, and able to manage a huge amount of sensor data streams. Hence, the system has been developed with a modern, solid, and easily extensible architecture. Furthermore, an open-source approach has been selected. Technical choices include the Python programming language Python (2009), the Django web application framework Diango (2009), the MySQL database management system MySQL (2009), and Weka Wek (2009), a Java-based machine-learning toolkit. The SeReNe prototype results in a Weka-based web application which can be easily exploited by any user through a web browser.

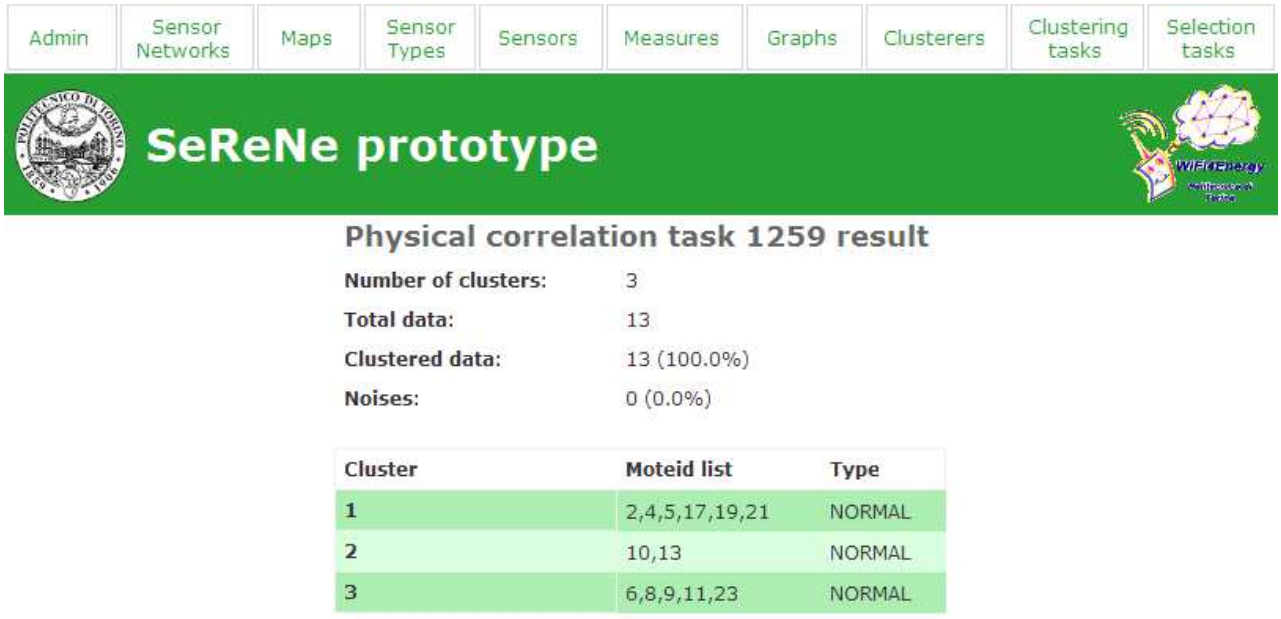


Figure 2: SeReNe prototype, clustering task result (example)

The SeReNe prototype allows the identification of energy-aware models for sensor network data by means of the following steps.

1. Performing correlation analysis on both spatial and temporal dimensions by dynamically exploiting Weka clustering algorithms.
2. Singling out the set of R-Sensors by means of the proposed strategies, which have been implemented in Python.
3. Identifying the most efficient transmission schedule for R-Sensors by exploiting Phyton-developed TSP solvers.

Finally, energy-aware sensor network models can be shown directly on the network map, providing the users with an intuitive visual feedback.

Different techniques proposed in literature to (i) deal with missing data (e.g., non-parametric Expectation Maximization techniques Davidson and Ravi (2005), association rules mining to detect likely value replacing erroneous measurements Jiang (2007)), and (ii) normalize data Jiawei Han and Micheline Kamber (2006) could be easily integrated in the SeReNe framework to enhance its capabilities in handling sensor data streams.

The SeReNe framework consists of two main components: a web-based front-end and a DBMS-based back-end. Details of SeReNe components are described in the following.

3.1 Front-end

The web application front-end consists of different tabs where the user can take the following actions.

- **Web site administration**, including user permission settings and raw access to the back-end database.
- **Sensor Networks** tab, allowing the definition of multiple sensor networks and their physical properties.
- **Maps** tab, where map images of the sensor networks can be uploaded.
- **Sensor types** tab, which allows to describe the different types of sensors available.
- **Sensors** tab, providing a list of sensors and their positions on the maps.
- **Measures** tabs, presenting a list of measurements for each sensor and time instant.
- **Graphs** tab, allowing to plot the measurements for specific sensors, measures, and time periods.
- **Clusterers** tab, where the available clustering algorithms are detailed.

- **Clustering tasks** tab, allowing to create and execute experiments, as well as present their results.
- **Selection task** tab, addressing the management of the experiments aimed at selecting the Representative-Sensors.
- **TSP** tab, addressing the management of the experiments aimed at computing the schedule which minimizes the communication cost and balances energy consumption among sensors by means of the Traveling Salesman Problem (TSP) solver.

The main features of the SeReNe implementation are detailed in the following, along with some sample screenshots of the web application prototype.

3.1.1 Sensor networks

The SeReNe framework allows end users to manage multiple sensor networks. Each sensor network is characterized by a name, a description, and a comma-separated list of physical properties, i.e., the measures which are sensed by the network. The list of the currently available sensor networks is presented in an intuitive tabular format. These data can be inserted by the user in a friendly web form, both when adding new sensor networks and when editing existing entries.

Sensor networks can be associated to map images in the Map tab. New map images can be uploaded through the web application and stored on the application-reserved folders on disk. During the upload, the user can choose to which sensor network the map has to be associated. Furthermore, scaling factors for both the x and y axis can be provided to convert the dimension of the map from pixels to meters. This conversion is useful because it enables the placement of the sensors on the map by expressing their coordinates directly using the meters as unit of measure (see Section 3.1.2). Finally, the sensor marker and its font sizes can be specified, thus allowing a fine-grained customization.

3.1.2 Sensors

The SeReNe framework provides two tabs for managing the sensors of a network. Different types of sensors can be created, edited and deleted on the sensor type tab. whereas the list of all the available sensors is shown in a different tab. In particular, added sensors are associated to a specific network, which is also a filtering criterion when displaying the sensors. Each sensor is given an identifier (unique in its network), a type (chosen among the previously entered sensor types), and a tuple with the coordinates along the x, y, and z axes. Coordinates are specified in meters with respect to a customizable reference point of the sensor network map, whose default value is the upper left corner. Since currently only bidimensional map images are supported, the x and y coordinates only are considered for plotting the results on the map.

3.1.3 Measurements

To drill down into the actual measurements to be analyzed, the SeReNe framework provides an interface that allows listing, filtering and sorting the raw values. Each measurement is given a globally unique ID, is associated to a specific sensor, has a timestamp, and consists of a list of (measure name, measure value) tuples. Measures can be removed or manually modified, mainly for testing purposes.

A more powerful interface is provided by the Graph tab, where the user can select a subset of measures to be plot. The plot can be limited by selecting one or more sensors, one or more measures, and a time range (possibly spanning multiple days).

3.1.4 Algorithms

The SeReNe framework is designed to dynamically load an external Weka Wek (2009) library, thus allowing SeReNe to automatically provide the user with the latest data mining algorithms. The library is loaded during the first startup of the application, but a reload can be requested if the library changes (e.g., it has been updated), without the need of restarting SeReNe. A list of all the available clustering techniques is presented in the Clusterer tab. For each technique, a name, a description, and the options are given, thus helping the user in her choice for a suitable clustering method.

3.1.5 Experiments

SeReNe models experiments by means of tasks. Each task is a single experiment: it is performed on a subset of the measurements (e.g., on a limited time period of interest, considering only one or more measures), it exploits a chosen algorithm with specific options, it focuses on a single sensor network, and it has its own results (or a

failure with reasons). Three main types of experiments are presented: (i) clustering tasks, (ii) selection tasks, and (iii) TSP tasks.

For instance, the Clustering task tab presents a list of performed experiments and corresponding options, gives access to their results, and allows the user to delete or create experimental tasks.

Creation of experimental tasks allows the user to choose among the available options. Furthermore, ranges for parameter values are accepted, along with a step value. This feature provides the user with an easy tool for creating multiple experiments, whose parameter values will be chosen in the given range and changed by the given step value.

A sample result is presented in Figure 2. The type of experiment and the task identifier are shown as page title, whereas result details are reported in a tabular format. Since this is a clustering task, details include the number of clusters, the number of analyzed and clustered samples, and the number of noise clusters (this is due to the specific clustering algorithm used in the experiment, DBSCAN).

Finally, a table with the actual clusters and the map of the clustering result are reported. In clustering result maps, sensors are identified by a circle next to their sensor-ID number, whose colors depend on the cluster they belong. The map shows the real location of the sensors in the laboratories of the Politecnico di Torino where experiments have been performed.

3.2 Back-end

The web application back-end can be any Python supported DBMS. A customizable database connection (host, port, database name, user name and password) can be configured at startup and required tables are automatically created. In our implementation, MySql 5.0 has been used with default settings.

Data stored in the back-end DBMS can be exported both using the web front-end and exploiting command-line scripts. Furthermore, it can be directly accessed by third-party applications. This features are especially useful for investigating experimental results by means of external tools, to archive or backup old data, and to improve the interoperability of the framework.

4 Experimental results

We validated the SeReNe framework by performing different experimental sessions focused on analyzing (i) the effectiveness in detecting sensor correlations, (ii) the accuracy of R-Sensors in representing the network state, and (iii) the effectiveness in reducing energy consumption.

Two sensor networks have been considered. In Section 4.1 the WSN deployed at the Politecnico di Torino in April 2009 has been used as real case study, whereas in Section 4.2 a publicly available dataset from the Intel Berkeley Research lab has been exploited to evaluate SeReNe.

Due to lack of space, a selection of experiments have been reported for both networks. Core experiments, including clustering of sensors and accuracy of R-Sensors, are reported for both networks. The Politecnico di Torino network features a detailed comparison of the clustering algorithms used for the physical correlation task, whereas for the Intel network, energy savings introduced by the TSP phase of SeReNe have been computed, thanks to the availability of transmission costs among sensors (which were unavailable for the other network).

Correlation analysis has been performed using different algorithms available in the machine-learning open-source environment Weka Wek (2009). Among them, DBSCAN showed to be a sensible default choice (see Section 4.1.2). Experimental results of the selection strategies highlight the effectiveness of querying representatives instead of the whole network, as reported in Sections 4.1.3 and 4.2.3. The error threshold provided by our model is always smaller than sensor accuracy.

Experiments have been performed on a PC equipped with an Intel Pentium 4 3.2 GHz CPU, 2 Gb main memory, Kubuntu 8.04 Linux operating system and Weka version 3.5.2.

4.1 Sensor network deployed at the Politecnico di Torino labs

The Politecnico di Torino WSN consists of 23 nodes, of which 15 have been actually included in the analyses. Excluded nodes have been affected by hardware failures and low-level problems which are beyond the scope of this paper. During the tests, a few more problems have been encountered, ranging from node thefts to temporary issues (e.g., unexpected battery shortage). Hence, not all the 15 nodes have always been available for analysis.

Each node consists of a Tmote Sky module Tmo (2009b). The Tmote Sky (see Figure 3) is a low power wireless module for use in sensor networks. It features an IEEE 802.15.4 wireless transceiver with antenna, a USB connection, and integrates humidity, light, and temperature sensors.

All the nodes have been configured to transmit their daily measures in bulk at a certain time of the day. In particular, antennas were switched on for half hour at 5.30 pm each day. The half-hour period allowed us to

Figure 3: Tmote Sky wireless sensor module

avoid transmission losses due to drift in the mote clocks. The bulk transmission have been chosen instead of the real-time monitoring because it greatly improved the mote lifetime, which was up to a month on average. The switching of the antenna, however, affected the measurements: the temperature always showed a suddenly increasing trend during the half-hour period of antenna power up. Collected sensor measurements are publicly available upon request to the authors.

This Section is organized as follows. Section 4.1.1 describes the experimental setting, Section 4.1.2 presents the results of the physical correlation experiments with a comparison among different clustering algorithms, and Section 4.1.3 analyzes the effectiveness of exploiting R-Sensors instead of the whole network.

4.1.1 Experimental setting

We considered historical sensor data collected from April 1st to April 30th, 2009. The Tmote Sky modules collected temperature, humidity, and light values once every 15 minutes by means of the TinyOS platform Tin (2009). The x and y coordinates of sensors expressed in meters with respect to the laboratory map are also known. The location of the experiments consists of 7 consecutive lab rooms connected by a long corridor. We exploited the concept of epochs to introduce absolute time references. Epoch 0 is defined as the beginning of the measurement session, i.e., April 1^{st}, 2009, at 0:00, then 1 epoch has been set as equivalent to 60 seconds.

A preprocessing phase has been performed before each run of the experiments, consisting in a normalization step by means of a standard Weka filter (measurement values are normalized in the [0,1] interval). To provide better analysis capabilities, SeReNe allows the user to choose a range of values for each parameter. Then, a grid parameter evaluation is performed, by varying each parameter at a time in the given range with a user-defined step. Finally, the user is presented with the results of each combination of parameter values, thus allowing an informed choice of parameter estimation. If the user does not specify any parameter range, then default values are proposed.

4.1.2 Physical correlation

Detailed experimental results of the physical correlation analysis on the Politecnico di Torino network have been reported for the temperature measure.

To study physical correlation, each clustering session analyzes all measurements collected from all sensors at a given epoch. We have run a large set of experiments on the weekday, the weekend and the night time bands. Furthermore, different clustering algorithms have been compared. In particular, our analyses considered the K-Means Juang and Rabiner (1990) algorithm, as a representative of the partitioning approach, the DB-SCAN Ester et al (1996) algorithm, as a representative of the density-based approach, and the EM G. McLachlan and T. Krish (1997) technique, as representative of model-based approaches.

Before discussing different results of the clustering algorithm choice, the following reference trends have been highlighted.

- During weekday nights and early mornings (approximately until 9 am), sensors are grouped into different clusters according to the sensor physical positions, typically divided by lab.
- During weekdays, sensor data are grouped together into two main clusters: sensors in labs and sensors in the corridor.
- During holidays, sensors are grouped into different clusters according to the sensor physical positions.

Insights on these trends, tend to highlight a strong correlation with human activities and air-conditioning of the labs. During working times, i.e., weekdays, laboratories behave like one big group, whereas the corridor has a completely different behavior. On the contrary, when there is no air-conditioning (i.e, night and holidays), each room tend to drift away in its own way, thus different clusters emerge for each laboratory besides the corridor.

Clustering results for different algorithms have been compared to the reference trends. The following default parameters have been used.

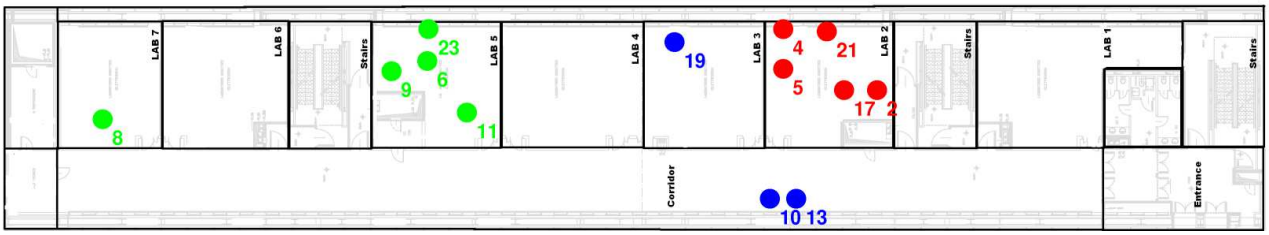


Figure 4: Physical correlation result, Sunday, April 5^{th}, 2009, 11 am, 6420 epoch, DBSCAN algorithm

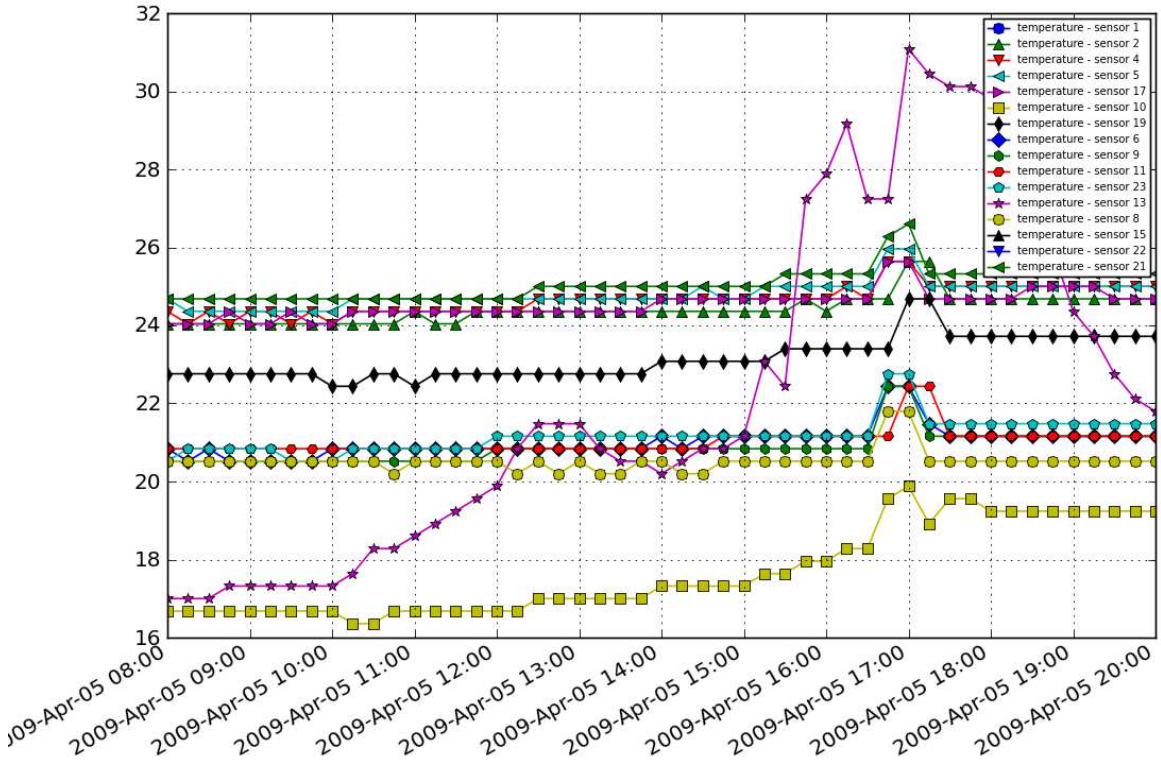


Figure 5: Overview of the temperature (°C) measurements on Sunday, April 5^{th}, 2009 for all the available sensors

- DBSCAN: epsilon 0.05, min points 2, euclidean distance (default built-in into Weka).
- Simple K-Means: number of clusters 2, random seed 100.
- EM: number of clusters -1 (means cross validation is used to select the number of clusters), max iterations 100, random seed 100, minimum allowable standard deviation 10^{-6}.

Results presented in the following figures consist of a map where sensors are indicated by a round mark, whose color depends on the cluster they belong to. The corridor covers all the lower (southern) part of the map, laboratories are the rooms located along the upper (northern) part. We have selected Sunday, April 5^{th}, 2009 as representative of the holiday results, and Thursday, April 2^{th}, 2009 as representative of the weekday results. At nights, results are very similar to those of holidays, apart from considerations on the sun light effects, as discussed later in this Section.

DBSCAN results, presented in Figure 4 and Figure 6, show that this algorithm follows the reference trends. During weekdays, the most common result consists of one main cluster with all the sensors in the laboratories and a second cluster with sensors in the corridor. On Sunday (and holidays), instead, two main clusters of the eastern and the western laboratories are identified, besides the corridor group.

DBSCAN also detects noisy sensors. The clusters consisting of three sensors, both on Sunday in Figure 4 (two sensors in the corridor and one in the center laboratory), and on the weekday in Figure 6 (two sensors in the corridor and one in the eastern laboratory), are marked as noise cluster.

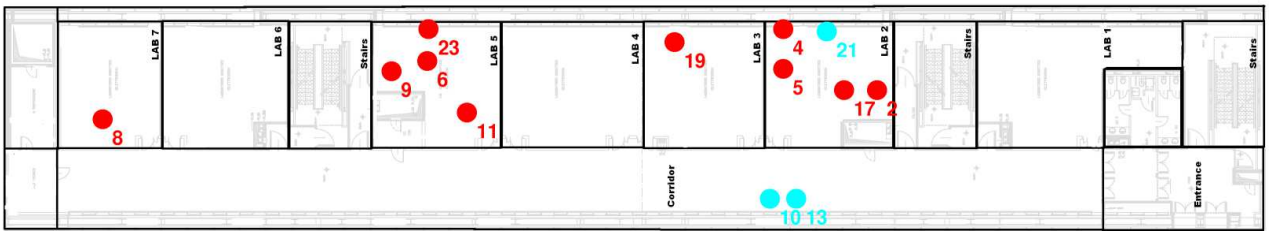


Figure 6: Physical correlation result, Thursday, April 2^{th}, 2009, 12 am, 2160 epoch, DBSCAN algorithm

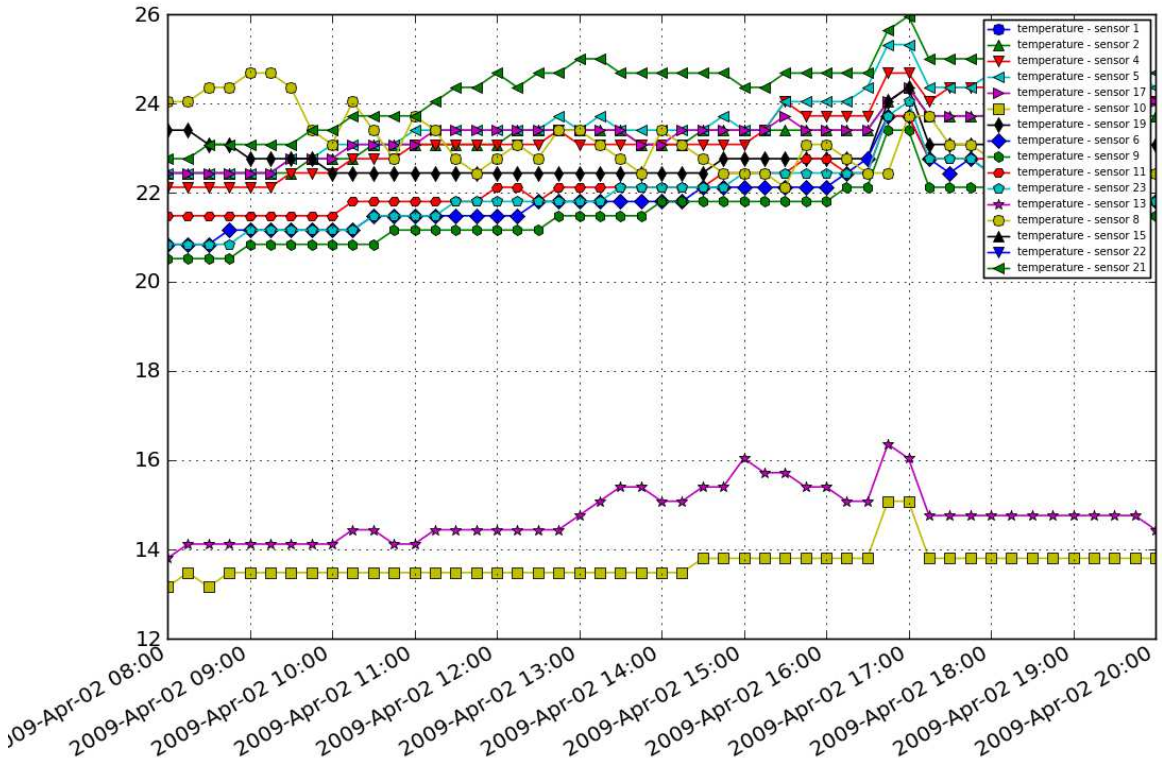


Figure 7: Overview of the temperature (°C) measurements on Thursday, April 2^{th}, 2009 for all the available sensors

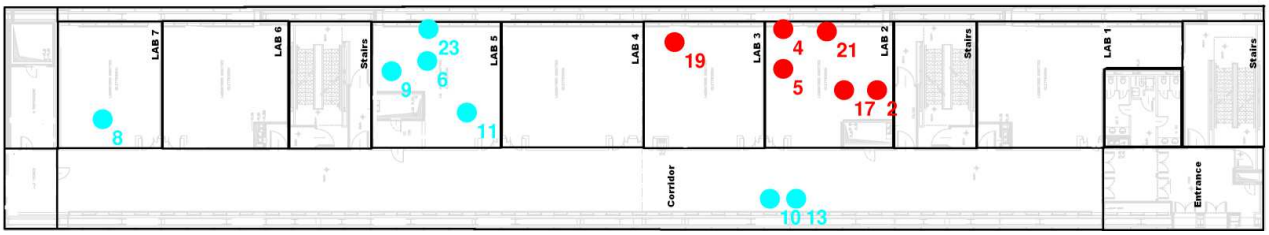


Figure 8: Physical correlation result, Sunday, April 5^{th}, 2009, 11 am, 6420 epoch, KMeans algorithm

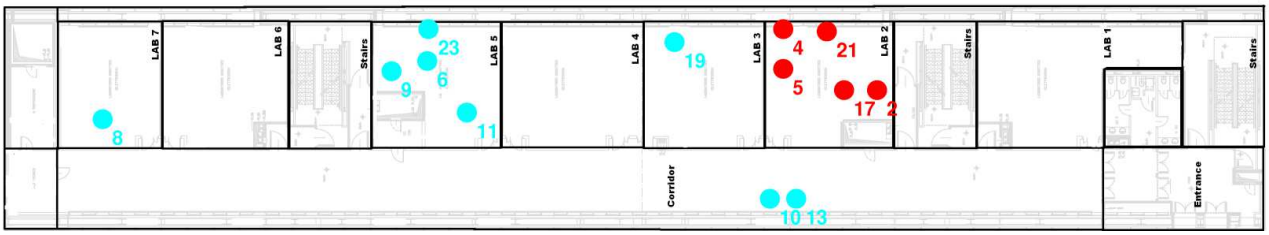


Figure 9: Physical correlation result, Sunday, April 5^{th}, 2009, 11 am, 6420 epoch, EM algorithm

A feature of this density-based approach is to label clusters as normal or noise. Noise clusters group values (i.e., sensors) that are not included in any other normal cluster. The details of the algorithm allowing to reach this results are beyond the scope of this paper, but results are interesting for analyzing possible reasons of noise-labeled sensors. In particular, we noticed that the couple of sensors in the corridor are typically grouped together in a normal cluster, except for a few cases, when they fall into a noise cluster. Further investigations revealed that one of the sensors, which is exposed to direct sunlight, reports a much higher temperature than the other, thus making both sensors fall into a noisy cluster. Other cases of noisy sensors are related to peculiar positions or conditions, e.g., in front of a cooling fan or affected by temporary human activities. In the case of the noise-labeled sensor in the middle laboratory on Sunday, its behavior is actually different from the other two groups. Looking at the raw measurements in Figure 5, we notice that the noisy lab sensor is the only one to gather measurements around 23°C, whereas the two main clusters of sensors are in the 24-25°C and in the 20°C. Measurements starting from 17°C in the morning are those of the corridor sensors. Notice how they diverge as the sun radiation affects only one of them.

Similar results have been obtained for the weekday, where a sensor of the eastern lab is behaving differently from the rest, thus being included in the noise cluster with the corridor ones. Evidence can be found in the raw measurement plot presented in Figure 7. The noise-labeled sensor reports temperatures constantly above 24°C from late morning, whereas other sensors gather lower values. The two lowest lines represent the temperature of the corridor (southern) sensors, which diverge along the day, as the sunlight effect increases.

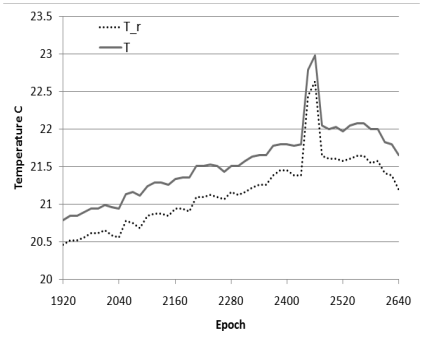
The capability of DBSCAN to report noisy data and its ability to divide the sensors into different numbers of clusters contributed to make it the default choice for the rest of the experiments.

KMeans results are presented in Figure 8. They are very similar to the EM results, which are shown in Figure 9. Both algorithms group together sensors in the corridor with sensors in the laboratories, even if analyzing the plot of the sensor temperatures (Figure 5), the corridor sensors have a completely different behavior. Proper tuning of the parameters sometimes improved this behavior, but differently from DBSCAN, no satisfying values could be found that consistently led to meaningful results for different days and times. Finally, notice that the DBSCAN noise-labeled sensor of the middle lab is grouped with different main clusters depending on the algorithm. KMeans puts it in the eastern lab cluster (see Figure 8), whereas EM groups it with the western lab and corridor sensors (see Figure 9).

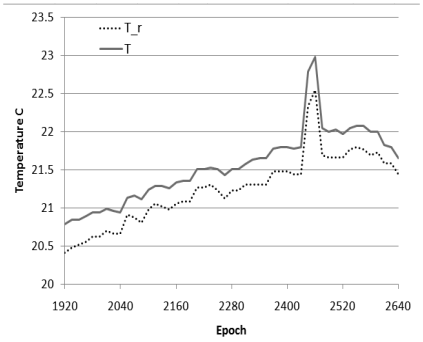
4.1.3 Accuracy of R-Sensors

To validate the effectiveness of exploiting R-Sensors instead of the whole network we analyzed: (i) Measurements collected by querying only R-Sensors with respect to querying the whole network, (ii) the accuracy of R-Sensors in representing the network state, (iii) the mean square error of our model in T_{model}, and (iv) the mean square error when we exploited the model to query the network during the same temporal interval of T_{model} in the following days.

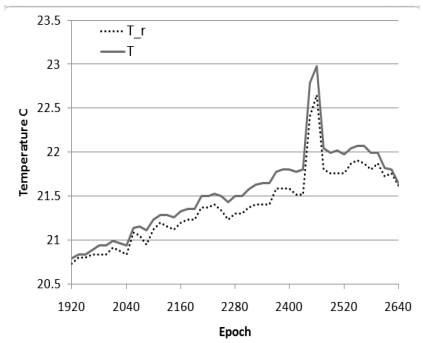
Experiments presented in this Section are based on the physical correlation results presented in Section 4.1.2 and obtained by exploiting DBSCAN. Hence, the same time periods have been selected. Three models have been used to analyze the sensor network. The first models the network during weekdays, the second during holidays, while the last during nights. Due to lack of space, results for the first two models have been selected



(a) Measure trend strategy



(b) Cluster shape strategy



(c) Cluster shape and core strategy

Figure 10: Average temperature querying only R-Sensors with respect to querying all sensors for the weekday model

to be reported in this paper.

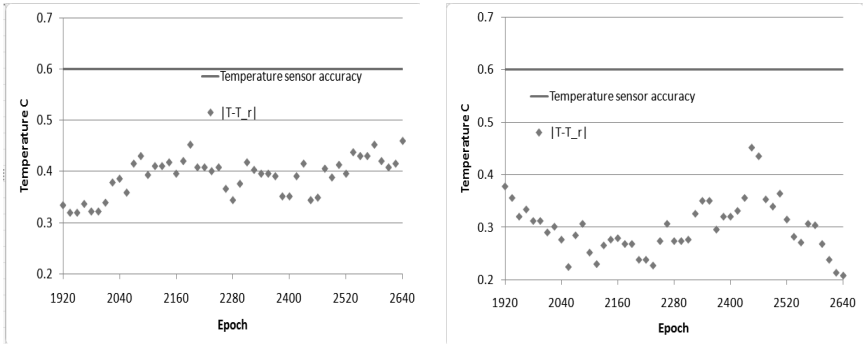
Sensor data collected during Thursday, April 2^{nd}, 2009 in 12 hours of monitoring, from 8:00 am to 8:00 pm, have been used as the training set for the weekday model, whereas sensor data collected during Sunday, April 5^{nd}, 2009 from 8:00 am to 8:00 pm, have been used as the training set for the holiday model. For the weekday sensor network model, representatives are selected from physically correlated sensor clusters related to the 2400 epoch, while for the holiday model representatives are singled out from physically correlated sensor clusters related to the 6600 epoch. After the correlation analysis, the best epoch window T_{model} for the weekday model ranges from 1920 to 2640 epochs (i.e., 12 hours, from 8:00 am to 8:00 pm), while for the holiday model ranges from 6240 to 6960 epochs. Hence, a model is able to represent the network state for weekdays and another for holidays. As default value, the number of representatives has been set to 50% of the network sensors. The generated sensor network model provides information for queries on both temperature and humidity measures, either independently or jointly.

For the three strategies (described in Section 2.2), Figure 10 shows the average temperature computed by querying only the representative sensors T_r and by querying all sensors T in different epochs in T_{model} for the weekday model. The three proposed strategies provide a good approximation of the monitored measures during T_{model}. The same conclusions can be drawn for the holiday model.

To validate the accuracy of the weekday model, Figure 11 plots the relative error (i.e., $\mid Tr - T \mid$) introduced by querying representatives instead of the whole network. The accuracy for the temperature sensor[1] is also plotted to detect when our model is characterized by an error lower than sensor accuracy. As shown in Figure 11 there are no measurements affected by a relative error above the sensor accuracy for every proposed strategy and for both considered models.

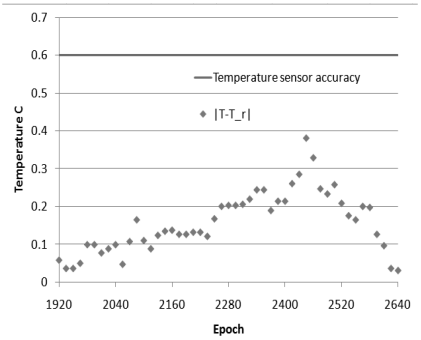
Furthermore, to estimate the error introduced by the proposed network models, the mean square error is computed in T_{model}, given by $\frac{1}{epoch\#} \sum_{t \in T_{model}} (\overline{M_{rt}} - \overline{M_t})^2$. $\overline{M_{rt}}$ is the average measure value computed by querying the representatives in a given epoch t and $\overline{M_t}$ is the value obtained by querying the whole network in the same epoch t. Figure 13 and Figure 14 show the mean square error on both temperature and humidity measures for the three proposed strategies in T_{model} (1920-2640 epochs) and T_{model} (6240-6960 epochs) by

[1]Data available on user manuals accessible at Tmo (2009a).



(a) Measure trend strategy

(b) Cluster shape strategy



(c) Cluster shape and core strategy

Figure 11: Relative error of the SN weekday model with respect to sensor accuracy: T_{model} 1920-2640 epochs

varying the percentage of representative sensors. For the temperature measure, all the proposed strategies provide an accurate model for weekday (i.e., T_{model} in 1920-2640 epochs) as shown in Figure 13(a), while for the humidity measure the second and third strategy provide a more accurate model than the first one, by leading to lower MSE values with few representatives (see Figure 13(b)).

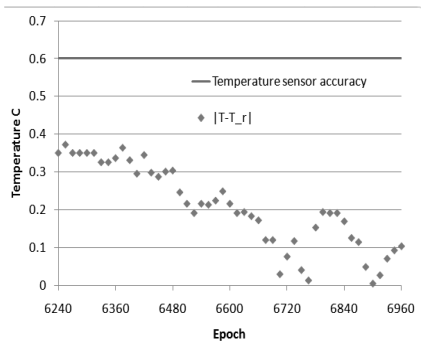
Note that the model built on the temperature data has been evaluated both on the temperature and humidity measures, and proved to be suitable in both cases, showing that it is actually more general than the single measure it was built upon. This is probably due to the correlation of such two measures.

For holidays (i.e., T_{model} in 6240-6960 epochs), all the proposed strategies provide an accurate model for the temperature measure, as shown in Figure 14(a), because MSE is always rather limited (i.e., smaller than 0.26). Instead, for the humidity measure the first and second strategy provide a more accurate model than the third one, by leading to lower MSE values with few representatives (see Figure 14(b)).

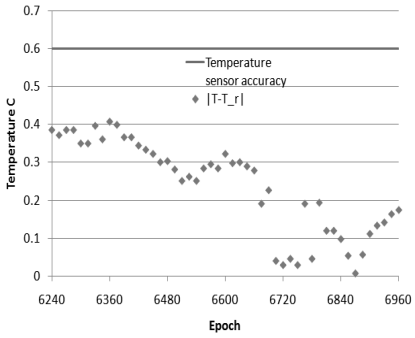
To validate the effectiveness of the proposed models on different time periods, the network has been queried during the same temporal interval on the following days. Representatives have been exploited in each epoch included in the time frame corresponding to T_{model}.

In particular, the weekday model has been exploited to query the network on Friday, April 3^{rd}, Tuesday, 7^{th}, and Thursday, 9^{th}, 2009, from 8:00 am to 8:00 pm. Figure 15, Figure 16, and Figure 17, show the mean square error on both the temperature and the humidity measures for each strategy on April 3^{rd}, 7^{th}, and 9^{th}, 2009 (on each graph) and with different percentages of selected representatives (on the x-axis). In all considered cases, the mean square error is comparable to the value obtained on the corresponding training data (see Figure 13). Hence, the identified weekday sensor model provides a good approximation of both the temperature and humidity phenomena.

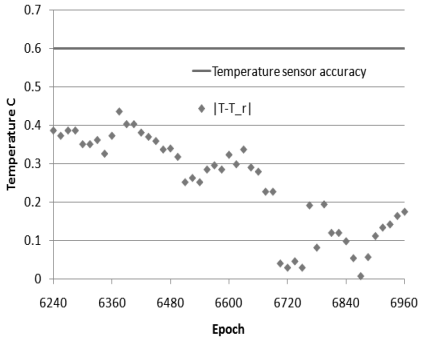
The holiday model has been exploited to query the network on Sunday, April 12^{th} and Sunday, April 19^{th}, 2009, from 8:00 am to 8:00 pm. Figure 18 and Figure 17 show the mean square error on both the temperature and the humidity measures for each strategy on the considered Sundays (on each graph) and with different percentages of selected representatives (on the x-axis). In almost all the considered cases, the mean square error is comparable to the value obtained on the corresponding training data (see Figure 14). The only exception is represented by the model with fewest representatives (i.e., 0.3%) exploited on April 12^{th}, 2009, which is characterized by a mean square error greater than the value obtained on the corresponding training data (see Figure 14). However, apart from this exception, the identified holiday sensor model provides a good



(a) Measure trend strategy

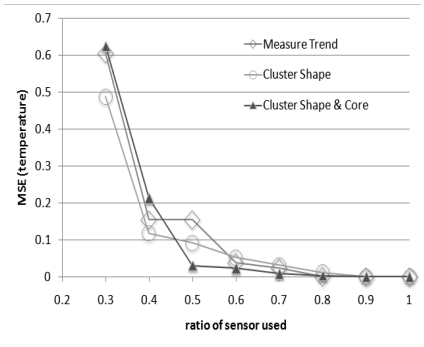


(b) Cluster shape strategy

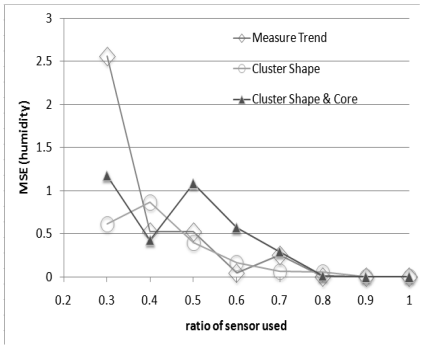


(c) Cluster shape and core strategy

Figure 12: Relative error of the SN holiday model with respect to sensor accuracy: T_{model} 6240-6960 epochs

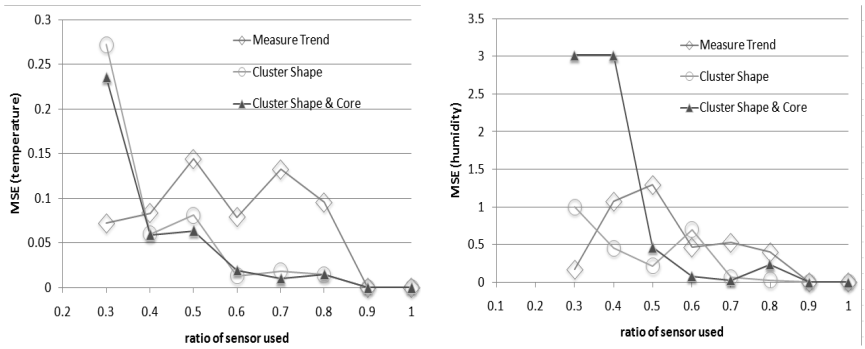


(a) MSE on the temperature measure



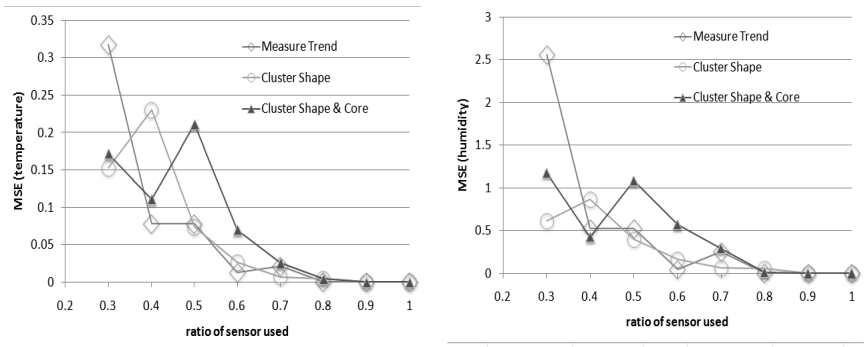
(b) MSE on the humidity measure

Figure 13: Weekday model: T_{model} in 1920-2640 epochs



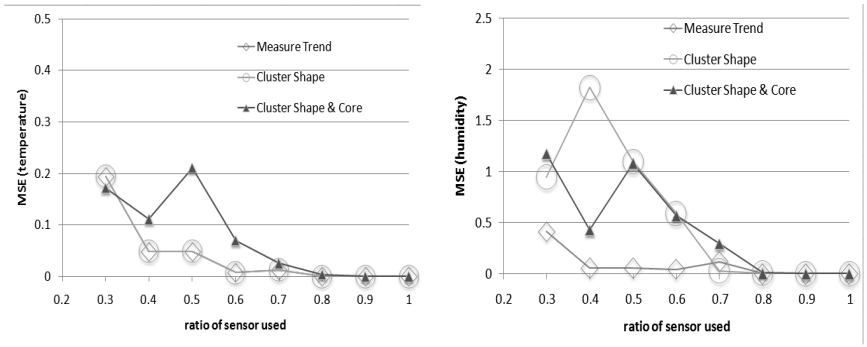
(a) MSE on the temperature measure (b) MSE on the humidity measure

Figure 14: Holiday model: T_{model} in 6240-6960 epochs



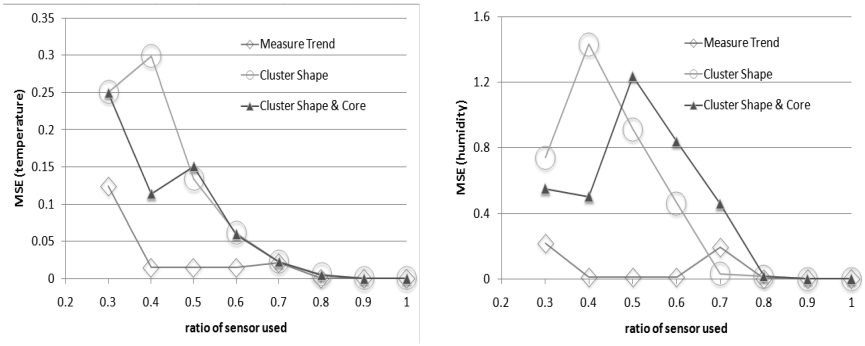
(a) MSE on the temperature measure (b) MSE on the humidity measure

Figure 15: Friday, April 3^{rd}, 2009, 3360-4080 epochs



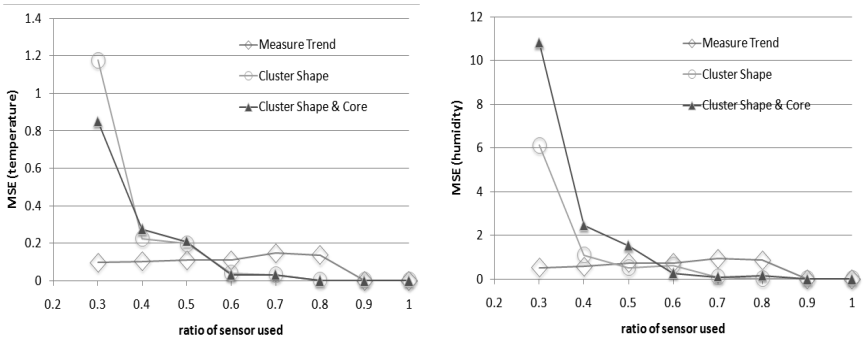
(a) MSE on the temperature measure (b) MSE on the humidity measure

Figure 16: Tuesday, April 7^{th}, 2009, 9120-9840 epochs



(a) MSE on the temperature measure (b) MSE on the humidity measure

Figure 17: Thursday, April 9^{th}, 2009, 12000-12720 epochs



(a) MSE on the temperature measure (b) MSE on the humidity measure

Figure 18: Sunday, April 12^{th}, 2009, 16320-17040 epochs

approximation of both the temperature and humidity measures.

4.2 Sensor network deployed at the Intel Berkeley Research lab

This Section is organized as follows. Section 4.2.1 describes the experimental setting, Section 4.2.2 presents the results of the correlation analyses, Section 4.2.3 analyzes the effectiveness of exploiting R-Sensors, and Section 4.2.4 investigates the energy saving by querying only R-Sensors instead of the whole network.

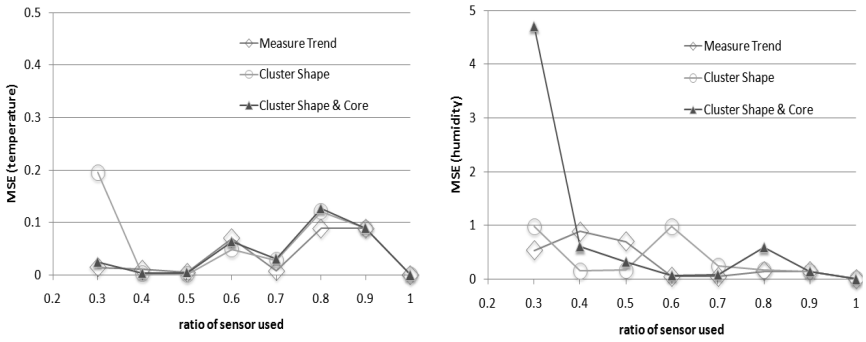
4.2.1 Experimental setting

We considered historical sensor data collected from 54 sensors deployed in the Intel Berkeley Research lab Intel (2009) between February 28th and April 5th, 2004. The dataset contains 2.3 million readings. Mica2Dot sensors collect temperature, humidity, light, and voltage values once every 31 seconds (epoch) by means of the TinyDB in-network query processing system Madden et al (2003), built on the TinyOS platform Tin (2009). The x and y coordinates of sensors expressed in meters with respect to the laboratory map are also known.

The analysis of historical sensor data is preceded by a preprocessing phase, which aims at smoothing the effect of possibly unreliable collected measurements. Preprocessing entails the following steps: (i) outlier detection and removal, and (ii) standardization. Faulty sensors may provide unacceptable values for the considered measures. We removed data outside the validity range for each measure (e.g., humidity<0 or humidity>100) and the entire sensor data when at least two measures had been reported to be unacceptable. After this preprocessing step, the dataset contained 1.7 million sensor data. Finally, for each remaining measurement, values are normalized in the [0,1] interval.

4.2.2 Correlation analysis

Correlation analysis of historical sensor readings analyzed both temporal and physical relationships. The first has been performed by considering separately each time series collected by each sensor. We performed the



(a) MSE on the temperature measure (b) MSE on the humidity measure

Figure 19: Sunday, April 19^{th}, 2009, 26400-27120 epochs

Table 1: Physical correlation result, Sunday, March 7^{th}, 2004, 6:40 pm, DBSCAN algorithm

Cluster	Sensor list	Type
1	18,19,20,21,22,25,26,29,31,32,33,53,54	NORMAL
2	23,24,27,34,36	NORMAL
3	35,37,38,39,40,41,42,43,45,46,48,49,50,51,52	NORMAL
4	14,16	NORMAL
5	44,47	NORMAL
6	9,10,11,17	NOISE

analysis for every combination of the collected measures (e.g., temperature, humidity, and light). Hence, different clustering sessions have been performed for each sensor data stream. For each session, the clustering algorithm returns a set of clusters. Each cluster is a set of correlated sensor values.

By plotting the clustering results on a mono-dimensional diagram where the x-coordinate represents time, we identified two/three cyclically repeated time bands, which always correspond either to daytime, or to night-time. The night band is shorter, and there is possibly an even shorter time band between day and night bands. Overlapped time bands, identified by different clustering sessions, are grouped together. For each group, the largest time band is considered to define the validity window T_{model}. Hence, the largest time band for the night-time and the largest time band for the daytime contribute to the corresponding ranges for T_{model}.

To study physical correlation, each clustering session analyzes all measurements collected from all sensors at a given epoch. Since some sensors could fail during the transmission or some values could be removed from the preprocessing step, some epochs report measurements for less than 54 sensors. We have run a large set of experiments on both the day and night time bands. For epochs belonging to the day time band, the following general trends have been highlighted: (i) During weekdays sensor data are grouped in a single cluster, which suggests that the lab is air-conditioned. (ii) During holidays, 3 or 4 main clusters are formed, depending on the epoch. (iii) For epochs in the night time band, 2 or 3 clusters are created depending on the epoch.

Figure 20 plots clusters on a Sunday at 6:40 pm. Inside the lab, three sub-areas with strongly correlated data can be identified, as well as two small clusters (two sensors) and a noise cluster with four sensors. Full experimental results are reported in Table 1.

4.2.3 Accuracy of R-Sensors

To validate the effectiveness of exploiting R-Sensors instead of the whole network we analyzed (i) the mean square error of our model in T_{model}, and (ii) the mean square error when we exploited the model to query the network during the same temporal interval of T_{model} in the following days, and (iii) the relative error distribution by querying only R-Sensors with respect to querying the whole network.

Sensor data collected during the night of February 29^{nd}, 2004 in 12 hours of monitoring, has been used as the training set for the model. For the night time band sensor network model, representatives are selected from physically correlated sensor clusters related to the 8685 epoch. After the correlation analysis, the best epoch window T_{model} ranges from 8524 to 9140 epochs (i.e., 5 hours and 28 minutes, from 11:57 pm to 5:25 am). As default value, the number of representatives has been set to 50% of the network sensors. The generated sensor network model provides information for queries on both temperature and light measures, either independently

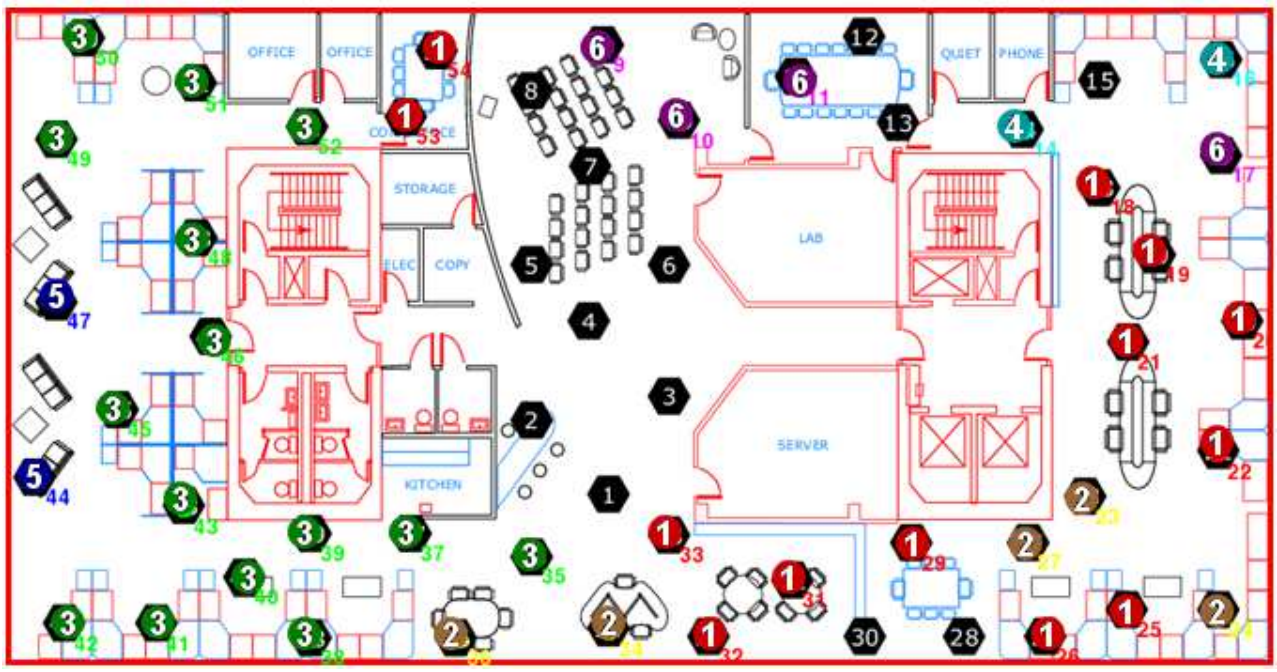
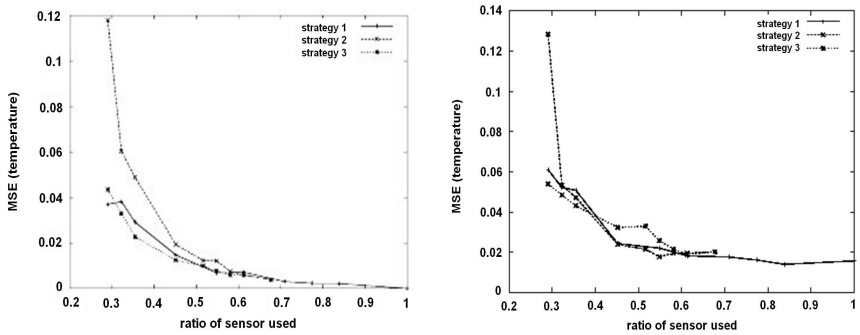


Figure 20: Physical correlation result, Sunday, March 7^{th}, 2004, 6:40 pm, DBSCAN algorithm



(a) T_{model} 8524-9160 epochs (b) 14287-14924 epochs

Figure 21: MSE on the temperature measure

or jointly.

To estimate the error introduced by the proposed network model, the mean square error is computed in T_{model}. Figure 21(a) shows the mean square error for the three proposed strategies in T_{model} (8524-9160 epochs) by varying the percentage of representative sensors. The first and third strategy provide a more accurate model than the second one, by leading to lower MSE values with few representatives.

To validate the effectiveness of the proposed model on different time periods, the network has been queried during the same temporal interval on the following days. Representatives have been exploited in each epoch included in the time frame corresponding to T_{model}. Figure 21(b) shows the mean square error for each strategy in a next day and with different percentages of selected representatives (on the x-axis). The mean square error is comparable to the value obtained on training data (see Figure 21(a)). Hence, the R-Sensors provide a good approximation of the monitored phenomenon.

Figure 22 shows the relative error distribution by querying only R-Sensors with respect to querying the whole network. The relative error distribution has been computed for all the proposed strategies and by considering values collected during the epochs in the range 14287-14924. Figure 22 also reports the error distribution related to a random election of representatives. All the four box plots have low median values (i.e., approximately zero). However, the inter-quartile range of the random selection strategy is the highest, proving that the error distribution variance of the proposed strategies is lower than a random choice. Even the worst distribution values of the proposed strategies are closer to the sensor accuracy (i.e., 0.2)[2] than a random selection. Hence,

[2]Data available on user manuals accessible at xbo (2009).

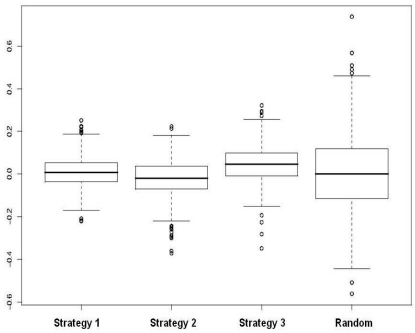


Figure 22: Boxplot - 14287-14924 epochs

the proposed selection strategies single out relevant representatives.

4.2.4 Energy Consumption

In this section we analyzed the energy dissipated by querying only R-Sensors with respect to the whole network. The total energy consumed by sensors is computed by considering (i) the total number of transmissions required by all R-Sensors and (ii) the energy consumed to transmit data. The latter is is obtained from the sensor data sheets (the radio link used on Mica2 motes and the Crossbow MTS400 Crossbow Inc. (2009) environmental sensor board).

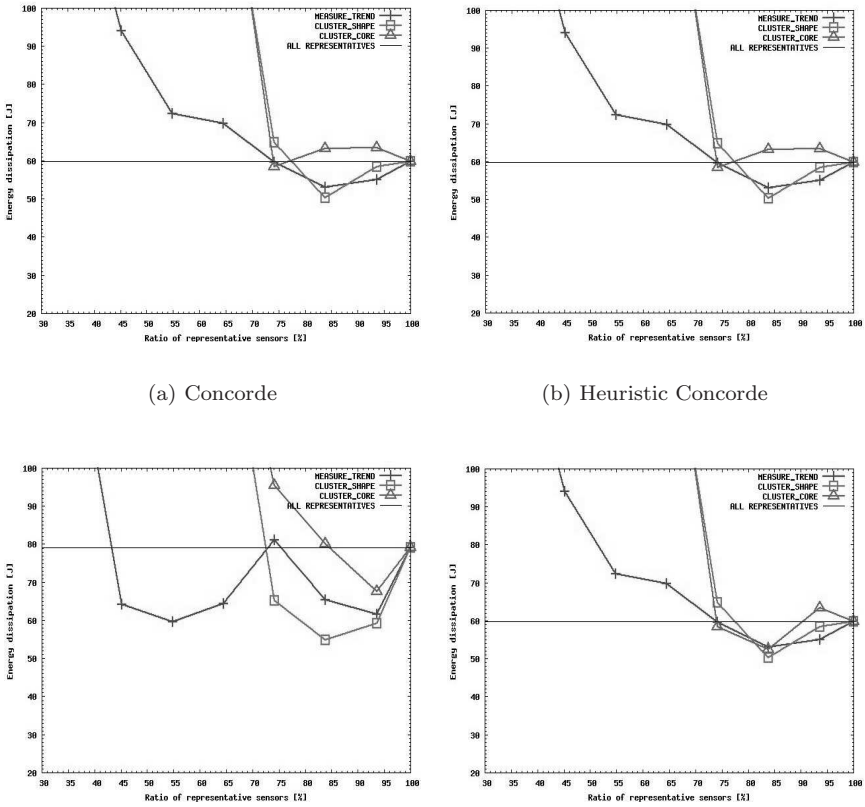
Figure 23 reports the energy dissipated by querying R-Sensors compared with querying the whole network. The transmission schedule has been identified by TSP solver algorithms integrated in the SeReNe framework. Experiments have been run by varying the ratio of selected R-Sensors. When the ratio of representative sensors ranges in 70%-95%, the energy consumption of R-Sensors is smaller than using all sensors. When a smaller subset of R-Sensors is selected, the communication cost significantly increases. This is mainly due to the decrease of the successful transmission probability. Since the number of retransmissions between two faraway nodes may be higher than that required among several close nodes, the corresponding energy consumption may increase.

Figure 23 also reports the energy consumption of each proposed strategy. For small percentages of R-Sensors (less than 75%), the measure trend strategy is able to select representatives whose usage saves energy during data collection with respect to the other strategies. For ratio values in the 70-95% range, all strategies lead to comparable energy consumption.

5 Related work

Many research activities in wireless sensor network have been devoted to identify careful power management techniques to efficiently query battery-powered sensors. Proposed strategies can be grouped in three classes: (i) Reduction in the number of transmissions, (ii) reduction in the number of sensors needed to answer a query, (iii) exploiting clustering algorithms to identify correlated sensors.

The first approach is based on reducing the number of transmissions needed to answer the query Deshpande et al (2004); Chu et al (2006); Deshpande et al (2005b); Tulone and Madden (2006). The focus was on a more robust interpretation of sensor readings. In Deshpande et al (2004) a statistical distribution of each considered phenomenon is independently inferred from the complete collection of sensor measurements. This model is then used to answer queries when the estimated accuracy is above a given threshold. Otherwise the query is redirected to (a subset of) the network, according to the required accuracy. Deshpande et al (2005b) proposed probabilistic models to capture correlations and statistical relationships among attributes collected by sensor devices. Network querying is performed by means of a pull-based approach to reduce communication cost during data collection. However, this technique does not react timely to network anomaly events. As proposed in Chu et al (2006) a push-based approach is more suitable for event detection applications. Ken Chu et al (2006) is a probabilistic model based on temporal and physical correlations. Since two dynamic probabilistic models are maintained, one over the network and another on a PC base station, anomaly events of the network are easily detected. The network is queried only when sensors do not sense values within an error bound. These approaches are efficient. However, neither of them works well when the topology of the network is dynamic. Authors in Tulone and Madden (2006) proposed the PAQ system (Probabilistic Adaptable Query system) which exploits time series forecasting for approximate query answering in sensor networks. The PAQ system allows to build a sensor network model in each sensor by guaranteeing a user-specified error threshold. Each network node performs a learning phase to build a local probabilistic model whose parameters are notified to the sink. The sink exploits all received models to predict the readings of individual sensors. When the sensor detects new



(a) Concorde

(b) Heuristic Concorde

(c) Genius Algorithm

(d) LKH Algorithm

Figure 23: Energy consumption: transmission schedules performed by different TSP solver algorithms

readings not represented by the local model (i.e., outlier readings), a re-learning phase is performed to build a new model, which is notified to the sink. Furthermore, the outlier readings are also transmitted to the sink. Thus, when the number of outlier readings increases, a large number of transmissions are required. To reduce the transmissions, more accurate local models are needed. However, accurate models require large learning windows to be stored locally in each sensor. Since sensors have limited resources, small learning windows are usually applied. The experimental results reported in Tulone and Madden (2006) have been performed on the sensor data from the Intel Berkeley research lab, which have been also considered in this paper. Only 5 sensors (i.e., 6,7,22,32, and 45) have been analyzed. For these sensors, the PAQ system is able to generate sensor model characterized (in the best case) by a median error between 0.017 e 0.03 for a user-specified error threshold set to 0.1.

The second approach is based on reducing the number of queried sensors. Kotidis (2005) first proposed to select a subset of sensor nodes to represent the network (i.e., a snapshot of the network Kotidis (2005)). For the election process, nodes need to exchange a set of messages with their neighbors to elect the representatives of the surrounding environment. The selection process is driven by spatial correlation discovered among sensor neighbors, thus only local similarities are considered. This approach has been enhanced by exploiting also temporal correlation among measures Silberstein et al (2006). However, none of the approaches is able to detect correlation among faraway sensors. Our approach is close to Kotidis (2005); Silberstein et al (2006). However, since we analyze historical sensor data, we do not require message exchanges among sensors to elect representative nodes. Furthermore, our approach is not restricted to sensor neighborhood when analyzing correlations.

The last approach is based on exploiting clustering algorithms to efficiently identify a subset of sensor nodes which best represent the network (e.g., the PREMON system Goel and Imielinski (2001), the LEACH system Heinzelman et al (2000), and the CAG technique Yoon and Shahabi (2007). PREMON system Goel and Imielinski (2001) performs energy-efficient monitoring based on a clustered architecture. Cluster-head nodes compute the prediction model by exploiting MPEG compression algorithms. Successfully predicted sensor data are not transmitted, thus, energy consumption is reduced. The LEACH system (Low-Energy Adaptive Clustering Hierarchy) Heinzelman et al (2000) assigns clusters based on the received signal strength and uses local cluster heads as routers. Furthermore, by means of a randomized rotation of local cluster-heads, the energy overhead among sensors is distributed through the network. The CAG technique (Clustered AGgregation al-

gorithm) Yoon and Shahabi (2007) discovers cluster of nodes by analyzing sensed measurements within a given spatial correlation threshold. This clustering remains valid as long as the sensor reading values are within a user-provided threshold (i.e., temporal correlation constraint). Only one sensor reading per cluster is transmitted, thus providing energy efficient acquisition and approximate aggregation of results within a user-provided error threshold. Any of the above approaches is efficient. However Goel and Imielinski (2001) requires the a priori knowledge of the cluster topology, Heinzelman et al (2000) is not able to detect correlations among faraway sensors, and Yoon and Shahabi (2007) transmits only one measure for each cluster. The SeReNe framework is more general with respect to the previous ones: (i) It does not require any a priori knowledge about cluster topology, (ii) it is able to detect correlations among faraway sensors, and (iii) it selects different R-Sensors from each cluster to better model the network state.

A prototype of the SeReNe framework was first introduced in Baralis et al (2007). The SeReNe framework, thoroughly described in this paper, enhances the preliminary work proposed in Baralis et al (2007) by significantly enriching the cluster sensor data analysis. Different clustering algorithms have been integrated in SeReNe to effectively discover both temporal and spatial correlations on both sensor data streams and sensors. Finally, this paper also presents and discusses an in-depth experimental validation of the SeReNe framework on two wireless sensor networks.

In the last few years, an increasing effort has been devoted by the researcher community to studying and designing efficient algorithms to discover correlations among sensor attributes. These correlations have been exploited to select the best execution plan to minimize the query execution cost. If two attributes are correlated, the execution plan always considers the attribute whose acquisition cost is lower Deshpande et al (2004, 2005a). Among data mining techniques, algorithms which mine frequent items Liu et al (2009) or itemsets Yang and Huang (2009) in data streams have been exploited to discover correlations among sensor attributes. Mining frequent items on sensor data allows the discovery of interesting and useful patterns among monitored measures (e.g., temperature and humidity). Algorithms extracting frequent items on data streams are presented and discussed in details in Liu et al (2009). Existing algorithms are grouped into sampling-based, counting-based, and hashing-based categories. Counting-based algorithms have been exploited in Ren and Guo (2009) to discover correlations among sensor attributes. However, to run counting-based algorithms on sensor data, the entire network has to be queried. Hence, efficient query execution strategies are needed also in this case.

A parallel effort in data stream analysis was devoted to the design of efficient indexing techniques to provide a good approximation of the monitored phenomenon. Authors in Luo et al (2009) proposed the SAO (Stream Approximate Onion-like structure) index for linear optimization queries against data streams. Both the index storage and the maintenance overheads are significantly reduced and experimental results, reported in Luo et al (2009), show the quality of the approximate answers. However, when the query is disseminated through the network, all sensors are queried.

6 Conclusions and future work

The SeReNe framework allows to efficiently discover optimized models for querying sensor networks while minimizing energy consumption for data collection. Given historical sensor data, two correlation dimensions have been analyzed by means of different clustering algorithms. Given a set of correlated clusters, a subset of representative sensors is singled out to optimally model the network state. Different TSP solver algorithms have been integrated into the SeReNe framework to select the best transmission schedule which minimizes the communication cost among sensors. The SeReNe framework has been tested on two wireless sensor network datasets. Experimental results demonstrate the adaptability and the effectiveness of the proposed approach in discovering energy saving models for sensor networks.

Future developments of this work will address the integration of distributed data mining algorithms and efficient turnover techniques for R-Sensors. Currently, clustering of sensor data is performed by exploiting different clustering algorithms which require the data to be stored in a single repository. Hence, the training phase involves a high number of transmissions from the sensors to the sink, to collect enough data for accurate sensor network model building. Furthermore, huge amounts of data require significant memory and good processing capabilities. To cope with data source distribution and to scale up data mining techniques, distributed data mining algorithms such as the distributed EM algorithm Safarinejadian et al (2011) may be exploited and integrated in the SeReNe framework. Furthermore, efficient turnover techniques for R-Sensors can also be promisingly integrated in the SeReNe framework. In particular, conventional scheduling policies such as the Round Robin algorithm or LRU policies may be exploited to this aim.

7 Acknowledgements

The authors would like to thank the research team of the WiFi4Energy project for technical contribution and Vincenzo D'Elia, Dante Bonino and Attilio Marchetti for developing parts of the SeReNe framework.

References

(2009a) Moteiv tmote datasheet. URL http://dbdmg.polito.it/~daniele/pub/tmote-sky-datasheet.pdf

(2009b) Moteiv tmote modules. URL http://www.moteiv.com/

(2009) The Weka Project for Machine Learning. Available: http://www.cs.waikato.ac.nz/ml/weka/

(2009) Tinyos. URL http://www.tinyos.net

(2009) Xbow user manuals. URL http://www.xbow.com/Products/productsdetails.aspx?sid=84

(2011) Concorde TSP Project. Available: http://www.tsp.gatech.edu/concorde/downloads/downloads.htm

Baralis E, Cerquitelli T, D'Elia V (2007) Modeling a sensor network by means of clustering. In: DEXA '07: Proceedings of the 18th International Conference on Database and Expert Systems Applications, IEEE Computer Society, Washington, DC, USA, pp 177–181

Chu D, Deshpande A, Hellerstein JM, Hong W (2006) Approximate data collection in sensor networks using probabilistic models. icde 0:48

Crossbow Inc (2009) Crossbow inc., wireless sensor networks. URL http://www.xbow.com/Products

Davidson I, Ravi S (2005) Distributed pre-processing of data on networks of berkeley motes using non-parametric em. In: SIAM SDM Workshop on Data Mining in Sensor Networks, pp 17–27

Deligiannakis A, Kotidis Y, Roussopoulos N (2004) Compressing historical information in sensor networks. In: SIGMOD, ACM Press, New York, NY, USA, pp 527–538

Deshpande A, Guestrin C, Madden SR, Hellerstein JM, Hong W (2004) Model-driven data acquisition in sensor networks. In: VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases, VLDB Endowment, pp 588–599

Deshpande A, Guestrin C, Hong W, Madden S (2005a) Exploiting correlated attributes in acquisitional query processing. In: ICDE '05: Proceedings of the 21st International Conference on Data Engineering, IEEE Computer Society, Washington, DC, USA, pp 143–154

Deshpande A, Guestrin C, Madden S (2005b) Using probabilistic models for data management in acquisitional environments. In: CIDR, pp 317–328

Diango (2009) Django website. URL http://www.djangoproject.com

Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp 226–231

G McLachlan and T Krishnan (1997) The EM algorithm and extensions. Wiley series in probability and statistics, John Wiley and Sons

Gehrke J, Madden S (2004) Query processing in sensor networks. Pervasive Computing 3(1):46–55

Goel S, Imielinski T (2001) Prediction-based monitoring in sensor networks: taking lessons from mpeg. SIGCOMM Comput Commun Rev 31(5):82–98

He T, Krishnamurthy S, Stankovic JA, Abdelzaher T, Luo L, Stoleru R, Yan T, Gu L, Hui J, Krogh B (2004) Energy-efficient surveillance system using wireless sensor networks. In: MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services, pp 270–283

Heinzelman WR, Chandrakasan A, Balakrishnan H (2000) Energy-efficient communication protocol for wireless microsensor networks. In: HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8, IEEE Computer Society, Washington, DC, USA, p 8020

Helsgaun K (2000) An effective implementation of the lin-kernighan traveling salesman heuristic. European J Oper Res 126(1)

Intel (2009) Intel lab data. URL http://db.csail.mit.edu/labdata/labdata.html

Jiang N (2007) A data imputation model in sensor databases. In: HPCC, pp 86–96

Jiawei Han and Micheline Kamber (2006) Data Mining: Concepts and Techniques. Series Editor Morgan Kaufmann Publishers, The Morgan Kaufmann Series in Data Management Systems, Jim Gray

Juang BH, Rabiner L (1990) The segmental k-means algorithm for estimating parameters of hidden markov models. Acoustics, Speech and Signal Processing, IEEE Transactions on 38(9):1639–1641

Kotidis Y (2005) Snapshot queries: Towards data-centric sensor networks. In: ICDE, pp 131–142

Liu H, Lin Y, Han J (2009) Methods for mining frequent items in data streams: an overview. Knowledge and Information Systems DOI 10.1007/s10115-009-0267-2

Luo G, Wu KL, Yu PS (2009) Answering linear optimization queries with an approximate stream index. Knowledge and Information Systems 20(1):95–121

Madden S, Franklin MJ, Hellerstein JM, Hong W (2003) The design of an acquisitional query processor for sensor networks. In: SIGMOD, ACM Press, New York, NY, USA, pp 491–502

MySQL (2009) Mysql website. URL http://www.mysql.com

Pang-Ning Tan and Michael Steinbach and Vipin Kumar (2006) Introduction to Data Mining. Addison-Wesley

Python (2009) Python website. URL http://www.python.org

Ren M, Guo L (2009) Mining recent approximate frequent items in wireless sensor networks. In: FSKD '09: Proceedings of the 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery, IEEE Computer Society, Washington, DC, USA, pp 463–467, DOI http://dx.doi.org/10.1109/FSKD.2009.607

Safarinejadian B, Menhaj MB, Karrari M (2011) A distributed EM algorithm to estimate the parameters of a finite mixture of components. Knowledge and Information Systems DOI 10.1007/s10115-009-0218-y

Silberstein A, Braynard R, Yang J (2006) Constraint chaining: on energy-efficient continuous monitoring in sensor networks. In: SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp 157–168

Skiena SS (2008) The Algorithm Design Manual. Springer; 2nd edition

Szewczyk R, Osterweil E, Polastre J, Hamilton M, Mainwaring A, Estrin D (2004) Habitat monitoring with sensor networks. Commun ACM 47(6):34–40

Tang C, Raghavendra CS (2004) Compression techniques for wireless sensor networks. In: Wireless sensor networks, pp 207–231

Tulone D, Madden S (2006) Paq: time series forecasting for approximate query answering in sensor networks. In: In EWSN, pp 21–37

Yang B, Huang H (2009) TOPSIL-Miner: an efficient algorithm for mining top-k significant itemsets over data streams. Knowledge and Information Systems DOI 10.1007/s10115-009-0211-5

Yao Y, Gehrke J (January 2003) Query processing in sensor networks. In: CIDR, ACM Press

Yoon S, Shahabi C (2007) The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks. ACM Trans Sen Netw 3(1):3