Exploiting Local Node Cache in Top-k Queries within Wireless Sensor Networks

Johannes Niedermayer*
Institute for Computer Science
LMU, Munich, Germany
niedermayer@cip.ifi.lmu.de

Mario A. Nascimento
Dept. of Computing Science
University of Alberta, Canada
mn@cs.ualberta.ca

Matthias Renz
Institute for Computer Science
LMU, Munich, Germany
renz@dbs.ifi.lmu.de

Peer Kröger
Institute for Computer Science
LMU, Munich, Germany
kroegerp@dbs.ifi.lmu.de

Hans-Peter Kriegel
Institute for Computer Science
LMU, Munich, Germany
kriegel@dbs.ifi.lmu.de

ABSTRACT

Top-k queries are a popular type of query in Wireless Sensor Networks. Typical solutions rely on coordinated root-to-nodes and nodes-to-root messages and on maintaining filters at the nodes, aiming at suppressing unnecessary messages, hence saving energy and furthering the network's lifetime. In this paper, we exploit the capability of a sensor node to cache a few recently observed values in order to determine "trends" for the observed values. Those trends can be used to further restrict the number of messages that need to be exchanged in the network, thus ultimately extending the network's lifetime. We compare our approach to the most recently proposed solutions in the literature using real and synthetic datasets, and we show that our approach is robust with respect to a variety of parameters and is able to improve the network's lifetime by up to 28% without any loss in the quality of the answer.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Databases*; H.2.4 [**Information Systems**]: Systems—*Query Processing*

Keywords

Wireless Sensor Networks, top-k queries

1. INTRODUCTION

Wireless sensor networks are usually defined as large-scale, wireless, ad hoc, multi-hop unpartitioned networks of homogeneous, small, static nodes deployed in an area of interest [9]. A single sensor node consists of one or more sensors, e.g., for temperature, acceleration, and light intensity, in combination with a microprocessor, a small amount of memory and a radio transceiver. By using these components, a node can take measurements of its surroundings, derive further information from the collected data, and send the resulting values to a root node via a path of adjacent nodes.

Applications for such kinds of wireless sensor networks include monitoring volcano activity [12], determining earthquake damage on buildings [8], and habitat monitoring [6].

Although already in use, sensor networks still show some deficiencies. The most important one is their limited amount of energy, because sensor nodes usually rely on battery power. Since networks are often deployed in hard to reach terrain empty batteries can not be simply recharged. Hence, the nodes' energy consumption must be reduced as much as possible in order to increase the network's lifetime. The most important factor when thinking of battery lifetime is the number of messages a node has to send/receive as well as their sizes. Therefore, the number of transmitted messsages and the length of a message should be reduced.

Because requesting every node's value is expensive in terms of energy consumption it is usually a good idea to keep track of a subset of nodes in a sensor network which contains sufficiently useful information, for example the median, quantiles or the k highest sensor measurements. The latter gives a good overview over extremes and can be used, e.g., for detecting anomalies.

In this paper we focus on top-k queries in hierarchical wireless sensor networks where the term *hierachical* refers to the structure of the underlying network which forms a routing tree. We will concentrate on computing the k largest values (modifications for computing the k smallest values are trivial). A continuous top-k value query computes periodically a set K containing the k largest values and their corresponding nodes from a set of nodes N, i.e. $K = \{(n_{i_1}, v(n_{i_1})), ..., (n_{i_k}, v(n_{i_k}))\}$ with $n_{i_j} \in N$ where $v(n_i)$ denotes the current measurement of node n_i.

In a straightforward solution for the top-k query, every intermediate node of the routing tree will collect all values from its children and forward the k largest of these values together with their corresponding node IDs. This approach is known as TAG [5]. Nonetheless there exist other sophisticated approaches like FILA [13] and EXTOK [7] that focus on reducing the overall number of transmitted messages

*Research performed while visiting the University of Alberta.

when performing a top-k query. Our main contribution in this paper improves on both of these algorithms.

The structure of the paper is organized as follows. First we present a short introduction to FILA, a related algorithm for solving top-k queries which relies on filters like EXTOK, and, more importantly, EXTOK which provides a basis for our contribution, T-EXTOK (Trend-EXTOK). Both FILA and EXTOK provide useful ideas towards T-EXTOK and therefore the improvements are motivated during the introduction of these solutions in Section 2. This will facilitate the understanding of T-EXTOK in Section 3 and 4. In Section 5 we present the results of a performance analysis we executed using both real and synthetic datasets. The paper is concluded by providing an overview of related work and future research in Section 6 and 7.

2. BACKGROUND

In order to present an overview over the existing solutions to address the problem of top-k queries in wireless sensor networks, two algorithms, FILA and EXTOK are introduced. Figure 1 shows example setups for each algorithm, using $k = 2$, at the beginning of each round.

Both algorithms consist of two phases. The initialization phase is performed once and returns the first top-k set while the update phase is performed as often as necessary to update the current top-k set at the root. The update phase is subclassified into three stages, data collection, refinement and filter broadcasting. We refer to the execution of these three stages together as a *round*.

EXTOK and FILA use filtering intervals to avoid unnecessary updates, i.e., nodes only have to send update messages if their observed value breaks the node's filter. In FILA, every top-k node owns its own filtering interval while all non-top-k nodes share a default interval. EXTOK uses only a default interval while recent top-k nodes send updates during each round.

2.1 FILA by example

During the first phase of the algorithm every node sends its value and ID as a tuple to the root, similar to TAG, but instead of k tuples k+1 tuples are forwarded because the k+1-th value is used to compute the upper bound of the default filter. The messages transmitted during initialization can be extracted from Figure 1(a). From the received tuples, in this case the set $\{(16, H), (19, D), (26, G)\}$, the root node computes the first set of top-k nodes and computes the first set of filters. The filters are created in a manner that filtering intervals do not intersect, the current value of a node lies within the bounds of its filtering interval, and all non-top-k nodes share the same filter. A simple solution to create a set of filters with these properties is to compute the midpoints between two consecutive values and use these midpoints as filter bounds for the nearest greater and the nearest smaller node. Besides the node with the highest value gets an upper bound of ∞ and the default filter for all non-top-k nodes gets a lower bound of $-\infty$. These filters are broadcasted to their respective owners if they change. Therefore, the top-2-nodes D and G receive the intervals $[17.5, 22.5[$ and $[22.5, \infty[$ while all other nodes use a default interval $]-\infty, 17.5[$.

During the next round, Figure 1(b), only nodes that break their filtering interval have to send update messages, in our example this is node H. The root node receives the new value of H and realizes that the value falls in the filtering interval of node D. Because the root node can not decide which nodes form the new top-2 set, it has to probe D which returns $(D, 18)$. After probing it creates the new top-2 set containing H and G and computes the new filtering interval for H, which is $[20, 22.5[$. Although 22.5 is not the midpoint between G and H, the filter of node G is not changed because only intervals of top-k-nodes whose value was updated during the current round are updated. The default filter is updated to $]-\infty, 20[$ because it must contain all non-top-k values.

In the second round, Figure 1(c), D has to send an update because its value moved outside the filter. H is probed because D fell into its filtering interval and the new top-2 set can be computed. Note that although H dropped below the top-2 threshold which was 22 in the end of the last round, the tree does not need to be fully refined.This observation will become important when looking at the functionality of EXTOK.

2.2 EXTOK by example

Equivalent to FILA, each EXTOK node has to send its value and ID during initialization because there is no information about the distribution of values available and every node could be a top-k node. Therefore, the first phase makes use of a TAG-like aggregation as shown in Figure 1(d). Again, note that node C does not forward all tuples but only the tuples with the highest 2 values, in our case $\{(19, D), (26, G)\}$, and therefore performs an in-network aggregation. From these tuples the root node computes the initial set of top-2 values and broadcasts the threshold $\tau_0^2 = 19$, which separates top-2 nodes and non-top-2 nodes, to all of its children (the subscript of τ denotes the current round while the superscript shows the value for k since thresholds depend on k). After each subsequent round t, τ_t^k is used by every node to decide on its current state. Nodes which value was greater or equal to τ_t^k during the according round become temporarily monitoring (TM) nodes, they will always have to send their new measurement and ID during the next round. In our example the nodes D and G become TM-nodes. All other nodes become filtering (F) nodes. F-nodes will only have to send a new tuple during round $t+1$ if their value grows larger than a specified separator σ_{t+1}^k. EXTOK uses the recently broadcasted threshold as the separator, i.e. $\sigma_t^k = \tau_{t-1}^k$.

Figure 1(e) shows the state of the sensor network in the beginning of round 1. Again, TM-nodes are drawn in a light grey. Nodes D and G have to send their value because they are TM-nodes. Beside these nodes, node H measured a value greater than $\sigma_1^2 = \tau_0^2$ and therefore has to send its value too. Therefore, the root always receives all values greater or equal to σ_t^k, possibly combined with some TM-node values less than σ_t^k. Because in this case the root received enough values greater than σ_1^2 it computes the top-k values from this set and broadcasts the new threshold $\tau_1^2 = 22$. G and H become TM-nodes. This is a very good case for the algorithm because no refinement is necessary. Infact, with our first improvement over EXTOK we will focus on increasing the probability of falling into this low-cost case by carefully reducing the value of σ_t^k.

During round 2, Figure 1(f), only the current TM-nodes send their updates and the value of node H falls below $\sigma_2^2 = \tau_1^2$. Therefore the root does not receive enough values greater than τ_1^2, and although the overall number of received values is equal to k it can not be sure that there are

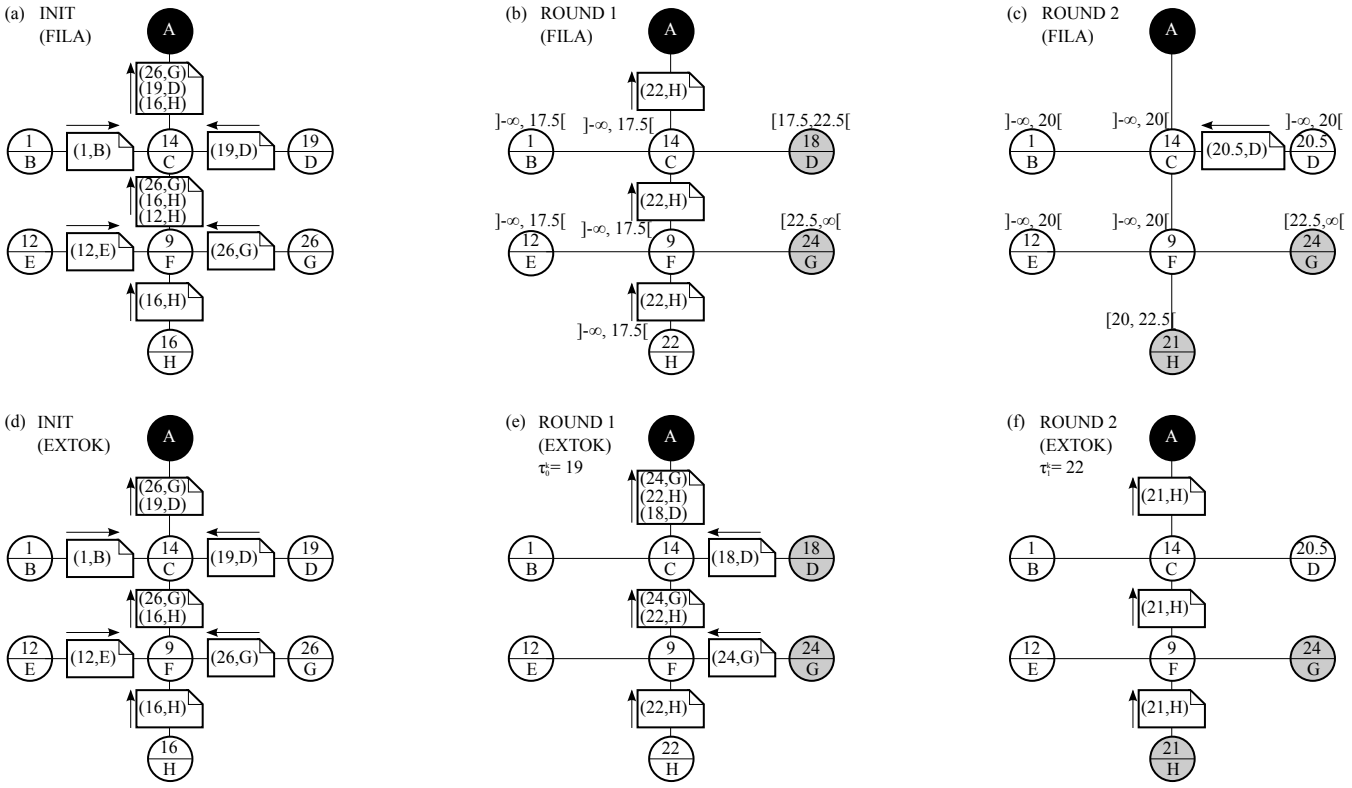

Figure 1: An example network showing the functionality of FILA (top) and EXTOK (bottom). The recent top-k (k=2) nodes are drawn in a light grey color, the root is visualized by a black circle. The upper half of a node shows its value while the lower half displays its ID. Messages between connected nodes are displayed in form of a rectangle, the direction of the message transfer is shown by an arrow beside each message.

no F-nodes in the interval $[21, 22[$ since F-nodes with values smaller than 22 do not send updates. Hence EXTOK has to probe the whole tree for values in the interval $[21, 22[$, although there is no relevant unknown node in this interval. This is an interesting observation, because by slightly reducing the separator we would have been able to reduce the number of message transmissions. Besides, FILA already performs this reduction of the separator by using the midpoint between the k-th and k+1-th value for the upper bound of the default interval. Beside this observation, EXTOK will send its new threshold in the end of this round. However, the top-k set did not change, therefore all nodes would have been able to derive their current state from their old state if the threshold would not have been broadcasted. We will cover this idea in Section 4.

The superior performance of EXTOK compared to FILA is due to two different reasons. First, even if the new threshold is greater than the recent threshold, it is possible in FILA that a refinement is necessary, for example if a top-k node's value falls into the interval of another top-k node's interval. This happens more often if measurements are inherently noise-prone and the noise exceeds the node's interval width. For this refinement, the number of transmissions is equivalent to two times the hop-count of this node to the root if there are no multiple refinement requests grouped together. In terms of transmitted values the root node must send at least the node ID while the child node has to respond with its current value and, if sending multiple requests in a single message, its node ID. Second, FILA sends different filters to the top-k nodes. Therefore the amount of data sent while broadcasting the new filters is greater than the amount of data sent by EXTOK's filter broadcasting because EXTOK sends the same threshold to all nodes.

Further details about EXTOK, including a proof of correctness, can be found elsewhere [7].[1]

The next sections introduce two improvements we suggested during the explanation of EXTOK and FILA, first an improved separator selection and then a new heuristic for broadcasting thresholds.

3. TO UPDATE, OR NOT UPDATE, THAT IS THE QUESTION

During each round, EXTOK decides which F-Nodes have to update their current value by comparing their value to the old threshold, i.e. $\sigma_t^k = \tau_{t-1}^k$. The last threshold is usually a good idea to use for aiding this decision because the probability for the new threshold being quite similar to the old threshold is quite high if the distance in time between two rounds is sufficient small. However the costs of performing a refinement if there are not enough values greater than the old threshold are high and in many rounds a refinement will be necessary.

[1]Since [7] has been accepted but not yet published at IEEE TKDE we made its pre-print available at http://bit.ly/9QQHWe for the convenience of the ACM GIS 2010 reviewers only.

To solve this problem it is useful to carefully reduce the separator and transmit more updates during the first update stage to avoid refining the whole tree. Although this increases the number of transmitted values during the first stage, we do most likely not have to do a refinement and therefore we can reduce the number of transmitted values and the number of message transmissions during the second stage. However reducing the value of the separator too much, e.g., to a value less than the smallest value in the sensor network would degrade EXTOK to a TAG-like behaviour because all nodes would have to send their values.

Therefore we developed and tested several heuristics that provide a tradeoff between number of updated nodes during the first stage and the probability of a refinement request during the second stage. We will concentrate on the best solution for the sake of this paper. This approach makes use of a regression model to predict the next threshold by using the information available from the m most recent thresholds. The computations can be performed at a node using minimal cache storage, therefore the root does not have to send more information; sending the new threshold like EXTOK is sufficient. Furthermore the computation of the separator is very simple, therefore the CPU load of a sensor node is only marginally increased.

It should be emphasized that with our separator selection we do not target on predicting the exact new threshold but rather on providing a pessimistic lower bound for the new threshold that can be used to reduce the probability of a refinement. Therefore, the optimality criterion for a separator σ_t^k which is used during round t is $\sigma_t^k \leq \tau_t^k$ and $\tau_t^k - \sigma_t^k$ should be minimized. τ_t^k is not known at the point in time where σ_t^k is computed.

Beside estimating the new threshold, T-EXTOK additionally reduces this predicted threshold by the maximum negative difference of all recent thresholds to the regression line in order to get a worst-case separator estimate. There exist several different models to predict a value from a set of existing values, but, in order to address the limited resources of a node by decreasing the computation complexity for nodes, we selected a linear regression model to estimate the next threshold. Therefore, we are looking for a function $\tau^{est}(t) = a + b \times t$ which minimizes the sum of the quadratic deviance of the last m thresholds to the linear function as explained, e.g., in [14]. The solution of this problem is the well-known formula $\tau^{est}(t) = a + b \times t$, where:

$$b = \frac{\sum_{i=t-m}^{t-1} (i - t_{avg})(\tau_i^k - \tau_{avg})}{\sum_{i=t-m}^{t-1} (i - t_{avg})^2}, \text{ and } a = \tau_{avg} - b \times t_{avg}$$

t_{avg} and τ_{avg} denote the average round count and average threshold, respectively, over the last m relevant values. By using this formula, we can compute an estimate of the next threshold $\tau^{est}(t)$. However, this function is still not sufficient because it is computed in a way that old thresholds lie beneath and above the regression line but we want them all to lie above, because, if assuming the same behaviour of the threshold in the future, this will make the next threshold lie above the regression line. Therefore, we have to subtract a term z from the resulting function that performs a downward shift. A good value for z is $z = \min_{i=t-m}^{t-1} (\tau_i - \tau^{est}(i))$ (note that the result of this formula is negative) which describes the maximum difference between all relevant old thresholds (with a value less than the respective value of the regression line) and their predictions. This term can be multiplied with a constant factor $c > 0$ to further improve the performance of the separator. In order to avoid that the separator becomes larger than the recently broadcasted threshold – which might be possible if we have a very fast upward trend, we compute the minimum of the predicted threshold and the last threshold. Therefore, the final formula for the computation of a separator can be written as

$$\sigma_t^k = \min(\tau^{est}(t), \tau_{t-1}^k) + c \times z$$

This separator is then used to determine which F-nodes have to send their values. Because $\forall t > 0 : \sigma_t^k \leq \tau_{t-1}^k$ the T-EXTOK result set of the first update stage will always be greater or equal to the EXTOK result set which reduces the number of refinements.

The example in Figure 2 visualizes this techniqe with a regression line that takes 5 measurements into account. The solid line visualizes the regression line. δ^- shows the maximum negative deviation of the last 5 thresholds from this regression line. Because the next predicted threshold is greater than the previous threshold, the previous threshold is used for further computation. From this we subtract $1.5 * \delta^-$ to compute the next separator ($c = 1.5$).

The selection of an adequate value for m is important for performance. If m becomes to large the regression line needs many rounds to react on short trends and rapid changes, but if it is too small, the regression line becomes very prone to noise. In our experiments we used a value of 5 for m which delivered good performance in general. For the first five rounds, the separator is selected equivalent to EXTOK because a node has not collected enough thresholds to compute the regression line at that time. For c we used a value of 1.5 during our experiments because this factor showed the best performance in our preliminary tests.

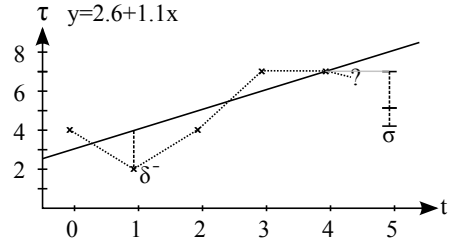

Figure 2: Regression driven separator selection.

4. IMPROVING BROADCASTED THRESHOLDS

Our first solution addressed reducing the number of refinements. This leads to significant improvements because broadcasting a message into the whole tree always requires $|N|-|L|$ messages where $|N|$ denotes the number of nodes in the tree and $|L|$ denotes the number of leaf nodes. However, there is another condition where we have to traverse the whole tree; broadcasting a new threshold creates the same number of messages. To reduce the number of messages, EXTOK only sends threshold updates if the old threshold

differs from the new threshold, i.e. $\tau_{t-1}^k \neq \tau_t^k$. When assuming trends in the measured data, this is usually not a good solution because the threshold increases or decreases over time. Nonetheless it is still very likely that a node that was a top-k node during round $t-1$ will still be a top-k node during round t. Therefore we decided to use a solution where the new threshold is broadcasted only if the set of top-k nodes changed. Note that the order of nodes in the top-k set can change between two consecutive rounds, the only condition is that a node that was a top-k node during round $t-1$ still has to be a top-k node during round t. The following lemma asserts the correctness of this solution.

LEMMA 4.1. *If each network node knows about its current state (TM-node or F-node) and it uses the same separator as the root node, our proposed approach will return the correct top-k set.*

PROOF. EXTOK has already been proven correct [7]. The case where the new threshold is broadcasted is equivalent to EXTOK because every node decides about its state by using the new threshold. Besides, T-EXTOK nodes can determine their current state by checking if the threshold broadcast was pruned. In this case the state of each node remains the same. If no threshold is broadcasted, the last broadcasted threshold will be assumed as equivalent to the new threshold, even at the root node. By using this information the new separator σ_{t+1}^k is computed. Equivalent to EXTOK, the root node will receive responses from all nodes with values greater or equal to σ_{t+1}^k during the next round, probably in combination with some TM-node values less than σ_{t+1}^k. This condition is equivalent to EXTOK, therefore the algorithm is still correct. □

This new solution increases the performance especially for small k because the probability that the top-k set will be formed from the same nodes as during the last round decreases with increasing k.

As discussed, we can reduce the number of transmitted messages during a round by $|N| - |L|$ if we use this approach. However, there emerge two new questions. First, F-nodes can only use outdated thresholds to decide if they have to send a new message. Does this imply a negative impact on the algorithms performance? Second, do these outdated thresholds yield any impact on the separator selection introduced during the last chapter?

To answer the first question, let us assume the following scenario. For simplicity we assume that $\sigma_t^k = \tau_{t-1}^k$ like EXTOK. Assume that during round $t-1$ the threshold of the top-k set was determined to be α. The top-k set changed and therefore the new threshold had to be broadcasted hence all nodes know the exact threshold of the last round during round t. During round t the set of top-k nodes did not change, however the top-k-th node's value dropped from α to $\beta, \alpha > \beta$. Because the new threshold β is not broadcasted into the network, the nodes do not know about this decrease in the threshold. They still use the old value of α to compute their new separator. Therefore if the value of a TM-node drops to $\gamma, \alpha > \gamma > \beta$ during round $t+1$ the whole tree has to be refined. We would not have had to do this refinement if each node had received the last real threshold β and therefore this refinement increases the number of message transmissions during this round by $|N| - |L| + |R \setminus (R \setminus U)|$, where R denotes the set of edges that have to be used to send the refinement response to the root node and U the set of edges that was used to transmit messages to the root during the first update stage. Therefore, $R \setminus (R \setminus U)$ contains all edges that were used during update *and* refinement. These edges would have been used twice and therefore lead to a performance reduction. The edges $R \setminus U$ would have been used once for sending messages to the root with both techniques, therefore they do not reduce the performance of the introduced algorithm. Because we saved $|N| - |L|$ messages during the last round the additional costs only total $|R \setminus (R \setminus U)|$. These costs usually correspond only to a small fraction of the costs $|N| - |L|$ emerging if a full tree traversal is performed. Nontheless it *is* possible to create cases in which we might loose performance. Consider a setting where top-k nodes always stay top-k nodes, i.e. the top-k set never changes. In this case, the new threshold would never be broadcasted and we would force a behaviour similar to TAG if there was an upward trend. To avoid this, another variable could be introduced that forces a threshold broadcast for example if the set of answers grows to small or to big. In this paper we do not explore this and leave it to future research.

The second question refers to the sensitivity of our threshold selection technique to this improvement. The separator selection relies on a continuous update of the threshold to adapt the new separator to the behaviour of the physical phenomenton. We observed that for small k the threshold broadcasting becomes more important and the performance of the regression-driven separator selection suffers, however the overall performance of T-EXTOK increases. For large k the improvement will be mostly due to the regression-driven separator selection and the threshold broadcasting algorithm becomes less important as already mentioned.

5. PERFORMANCE ANALYSIS

5.1 Test Setup

Given that energy supply is a severe constraint within a wireless sensor network, the network's lifetime is a useful metric to determine an algorithm's efficiency. As such, we compare the yielded network lifespan of T-EXTOK to those yielded by EXTOK and IFILA, using both synthetic and real datasets. Furthermore the number of transmitted values and the number of message transmissions effects the network lifetime so that the influence of these two factors are inherently included in our performance analysis. In order to determine the network lifetime, we performed 10 simulation runs for each tested algorithm until the first node of the corresponding sensor network ran out of power. All compared algorithms used the same physical and logical network topology during a simulation run. The topology was changed between two consecutive simulation runs. On real world data sets the topology was only changed by selecting another root node because the node positions were already defined. The synthetic dataset also enabled the possibility to reposition the nodes between two simulation runs. We made sure that all algorithms used the same infrastructure during each simulation run.

5.1.1 Node Distribution

For performance analysis, nodes were distributed in a rectangular area according to the underlying dataset. After distribution, the physical neighbours of each node were com-

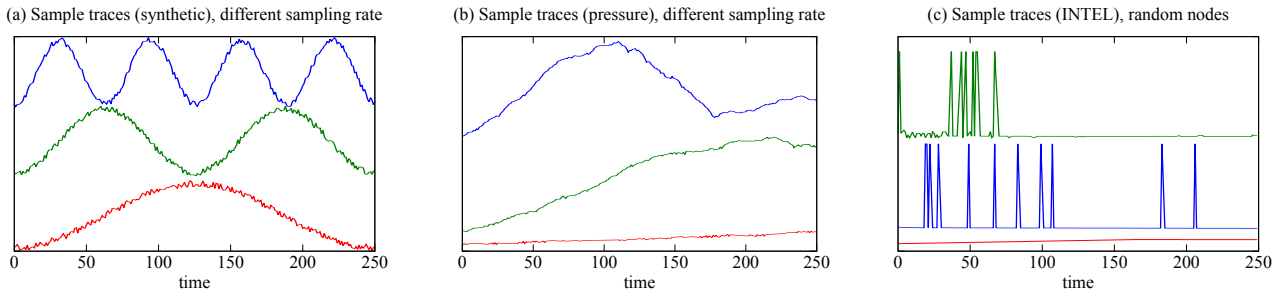

Figure 3: Sample traces of the three data sets. Synthetic and pressure data is plotted with different sampling rates. Because of its randomness the INTEL dataset shows traces of three different nodes. Graphs are stacked for sake of clarity, i.e. the y-axis is meaningless.

puted by finding all nodes in a speficied radius – the radio range d – in the neighbourhood of the node. This information was then used to create a routing tree by reducing the overall set of physical connections between nodes to a small subset of logical connections. For our simulations, we used a Dominating Set Tree due to its superior performance compared to a Shortest Path Tree [7]. For each node it is only possible to send messages to its children or to its parent. Although the original FILA paper assumes that all nodes can be reached by the root directly, we argue that even in this direction it is unlikely that there is not an obstacle interfering with the radio waves and therefore we assume a multihop setting in both directions, from/to the root.

5.1.2 Synthetic Dataset

The synthetic dataset was created on the fly when performing a simulation run. Nodes were distributed in an area of 200m in width and length. For initialization of the values we used an image containing interpolated noise (Figure 4) to simulate the similarity of neighbouring values in reality. Each node's position in the 200mx200m area was mapped to the corresponding coordinates in the picture and the resulting greyscale value was scaled to a range of $[0, 1]$.

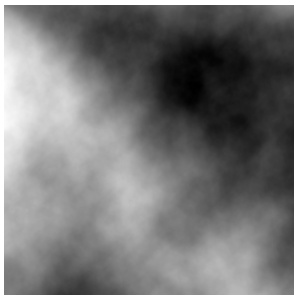

Figure 4: Initial value distribution when using synthetic data. Bright areas denote values similar to 1, dark areas visualize values similar to 0

Because the input image only produced 256 different values, we added some additional noise by using a Pseudo Random Number Generator. The noise magnitude was set to less than $\frac{1}{256}$ in order to keep the overall distribution of values in the image. This data was used to initialize the algorithms and compute the first set of top-k values. To compute consecutive sets of values, we used a sinoid function with additional noise. For the synthetic dataset we chose the sampling rate, the underlying algorithm, k, the number of nodes N, and the radio range d of a node as independent variables. In Figure 3(a) we show traces for the synthetic dataset with different sampling rates.

5.1.3 Real Datasets

We also used two datasets containing real data: the INTEL Berkley[2] dataset used by the EXTOK team and a dataset containing pressure data provided by the team that developed IFILA.

The INTEL dataset consists of 54 nodes, for each node there are 60000 measurements. The original coordinates of the nodes in the Berkley lab were given, therefore we only changed the ID of the root node between two simulation runs. In order to evaluate our proposal, when using this dataset we only changed d, k and the sampling rate.

The IFILA team had extracted data traces for ca. 1000 nodes from the Live from Earth and Mars project[3] at the University of Washington. However this dataset did not contain node positions. Because the IFILA team assumed locally clustered data during their test runs, we used a self-organizing map approach similar to [3] to sort similar values near together in an area of 200m in width and length. Feature vectors of size one were used as input of the self organizing map, containing the first measurement of each node. We received the position of each node as an output. By using this dataset, we explored the behaviour of all algorithms with changing sampling rate, k and d. Equivalent to IFILA we used 1022 nodes during our test runs.

An overview showing values for independent variables can be found in Table 1. In Figure 3(b) and (c) we show sample datatraces from the pressure and INTEL dataset respectively. The data trace for the pressure dataset is visualized by taking different sampling rates into account. The traces of the INTEL dataset just show different traces from three distinct nodes.

5.1.4 Cost Function

To determine the lifetime of a node within a sensor network, we use the same cost function as used for the original FILA [13] research. It distincts between three different modes: sending a message, receiving a message and sleep-

[2]http://db.csail.mit.edu/labdata/labdata.html
[3]http://www-k12.atmos.washington.edu/k12/grayskies/nw_weather.html

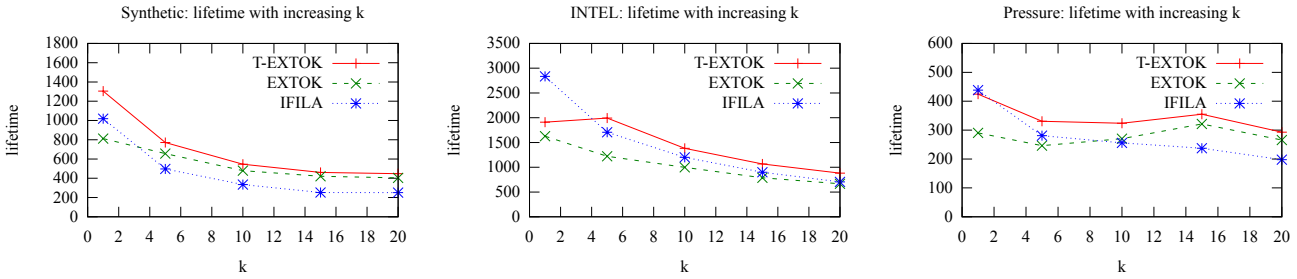

Figure 5: Effect of k on the network lifetime.

Table 1: Independent variables used during simulation. Bold text denotes default values.

Variable	Values
k	1, 5, **10**, 15, 20
N	100, 200, **300**, 500 (synthetic) **54** (INTEL) **1022** (pressure)
d	25, 30, **35**, 40, 45 (synthetic, pressure) 8, 10, **12**, 14, 16 (INTEL)
Sampling rate	250, **125**, 63, 32, 16 times/cycle (synthetic) every 1/**2**/4/8/16th value (pressure) every **1**/2/4/8/16th value (INTEL)

ing. The costs for sending a message is computed by the function $s * (\alpha + \beta * d^q)$ where s denotes the size of the transmitted packet, α is a distance independent constant, β is a distance dependend constant multiplied by d^q, with d the radio range of a node. Equivalent to FILA we assumed $\alpha = 50nJ/bit, \beta = 100pJ/bit/m^2$ and $q = 2$. The energy consumption for receiving data was computed by the function $s * \gamma, \gamma = 50nJ/bit$. For simplicity, we set the energy consumption in sleeping mode to 0. The initial energy supply of each node was set to 0.1 Joule.

We assume that a measurement and a node ID consists of 2 bytes, for sending the requested number of nodes - as necessary in EXTOK and FILA refinement requests - we assume 1 byte. Packet headers are taken into account as $s_h = 8$ bytes, the maximum packet size s_{max} is set to 128 bytes. Therefore, the maximum payload size s_p can be easily calculated. We decided about the magnitude of these variables by taking protocol specific constants from the 802.15.4 standard[4] into account. This standard is used for example by SUNSpot nodes and as a base for the ZIGBEE standard. However for the sake of simplicity we reduced packets to a header and a payload and did not account underlying techniques for collision avoidance and protocol specific behaviour.

When sending a packet the consumed energy is reduced from the sender's energy supply if the sender is not the root node because we assume an unlimited energy supply for the root. If a node sends a message, the receiving energy is subtracted from the remaining energy of all of its physical neighbours because we use a wireless medium and therefore a node has to receive a packet before it can decide whether or not it has to process it.

[4]http://standards.ieee.org/getieee802/802.15.html

5.2 Network Lifetime

In order to compare the network's lifespan yielded by the different algorithms we investigated how they are affected by the parameters listed in Table 1. We measured the number of rounds until the first node ran out of energy, and followed the traditional approach of keeping all parameters but one fixed at the default value, while varying the "free" parameter.

5.2.1 *Varying k*

When comparing the three solutions for different k (Figure 5) and using the synthetic dataset, T-EXTOK gains up to 28% compared to IFILA which stays alive for 1018 rounds. For pressure and INTEL data, T-EXTOK outperforms its competitors by up to 19% (EXTOK: 270 rounds) and 25% (IFILA: 704 rounds) respectively. On the synthetic dataset, T-EXTOK performs best for small k. However, its performance gain diminishes more and more with increasing k. This happens because hot spot nodes, for example routing nodes that have to forward a large amount of packets, will still have to forward update packets from their children. With increasing k, this part of the energy consumption becomes more and more important because the stress of hotspot nodes increases with increasing packet sizes and the overall probing of the network becomes less important. Because our algorithm mainly reduces probing of the network, our algorithm decreases energy consumption of every node, but this reduction is less important for hotspot nodes. However it is possible to alter the hierarchical structure of the network during consecutive update rounds because routing nodes do not store significant information about their children in EXTOK and T-EXTOK. This technique would change the set of hot-spot nodes which would increase network lifetime, and because T-EXTOK reduces the overall number of transfers it would positively affect T-EXTOK most. We note that we chose to not implement implement this feature in our experiments because we wanted to compare the "pure" behavior of T-EXTOK.

Another effect of increasing values for k is that the development of the top-k threshold becomes smoother as it can be seen in Figure 6 using the INTEL dataset. This observation has a positive effect on the performance of our solution because with a less fluctuating threshold the regression line achieves better results since its prediction is more similar to the new threshold. This behaviour can be observed when comparing the INTEL and pressure dataset with the synthetic dataset. With small k a performance loss of T-EXTOK can be observed. It becomes most obvious in the

INTEL dataset where the threshold is usually similar to 20, but sometimes increases to 120 during a round while decreasing to 20 again during the next round. Note that this is one of the worst things that can happen to T-EXTOK because the regression line approach is very prone to a short-lived change in this magnitude. However, with increasing k, T-EXTOK performs better than both competitors even under these circumstances.

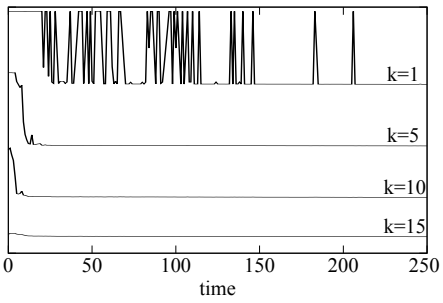

Figure 6: Development of the top-k threshold of the Intel dataset for different k, stacked.

5.2.2 Varying the sampling rate

Another interesting observation appears when comparing the performance of the different algorithms by taking the sampling rate into account (Figure 7). If the sampling rate of a physical phenomenon decreases, trends are more and more reduced to very abrupt changes. Therefore, the performance of an algorithm decreases with increasing sampling rate. The reason for this behaviour is that all solutions rely on some kind of filter. With a low samling rate the change in a phenomenon is usually more extreme than with a higher sampling rate. But if the width of filtering intervals stays similar this means that intervals will be broken more often with decreasing sampling rate. Because a broken filter always implies an update, the number of transfers and transmitted values increases which negatively affects the network's lifetime. This behaviour is most apparent for the pressure and synthetic dataset. In both cases, the performance decreases with decreasing sampling rate.

When making use of a linear regression model an approach might loose even more because the linear regression reacts slowly to trend changes, therefore the approximation of the next value becomes worse if the trend becomes too short-lived. This behaviour is clearly visible in the synthetic dataset, because the performance of T-EXTOK drops faster than the performance of EXTOK and FILA for low sampling rates.

If one choses a greater m (number of past observations) to compute the regression line even slower trends will lead to a performance loss. On the other hand, by reducing the length of the regression line, the performance loss can be reduced on fast trends. However a "short" regression line becomes very noise-prone. The INTEL dataset does not really show trends and therefore changing the sampling rate of the phenomenon does not show the expected effects.

During our experiments with different sampling rates, T-EXTOK gained up to 14% (IFILA: 463 rounds), 23% (IFILA: 270 rounds) and 30% (IFILA: 990 rounds) on the synthetic, pressure and INTEL datasets, respectively, compared to its strongest competitor.

5.2.3 Varying d

When increasing the radio range d (Figure 8), one might expect the network to last longer because the height of the network decreases and therefore less messages will have to be sent. This again decreases the number of transmitted values because the number of package headers is reduced.

However, the network lifetime of all approaches diminishes with increasing d. This is due to three circumstances. First, the number of children of a hot-spot node increases with increasing radio range. Therefore hot-spot nodes will have to receive more packets and send more data to their parents. Second, the overall energy consumption of the whole network for sending a message increases because more nodes will receive a message. And third, sending nodes need more energy to send a message due to the formula introduced in 5.1.4. All of these circumstances reduce the network lifetime. This behaviour is clearly visible in our test results, the performance of all approaches drops with increasing radio range.

With varying radio range, T-EXTOK gained up to 16% (EXTOK: 395 rounds), 22% (EXTOK: 402 rounds) and 16% (IFILA: 1250 rounds) on the synthetic, pressure and INTEL datasets, respectively, compared to its strongest competitor.

5.2.4 Varying N

Recall that he number of nodes was only changed when testing the synthetic dataset as the real datasets had a fixed number of nodes. The results can be found in Figure 9. For an increased number of nodes the performance decreases mainly because the number of intermediate nodes increases with k, although the height of the tree stays the same. Because the number of intermediate nodes increases, refinement and threshold broadcasting becomes more expensive. With varying node count, T-EXTOK gained up to 13% compared to its strongest competitor, EXTOK with 395 rounds.

5.2.5 Additional Factors Affecting Network Lifetime

Usually a sensor network is used to perform several concurrent queries, for example a top-k query for temperature and a top-k query for sound level. These two top-k queries do not necessarily overlap, i.e. the result sets for both queries will most probably be different. Now if we reduce the overal number of probes that traverse the whole tree for one query the energy supply for hot-spot nodes relevant to the other query will be conserved. Therefore the overall lifetime for a sensor network performing several different queries would benefit from T-EXTOK, because T-EXTOK focuses on reducing the number of refinements.

Another important factor when thinking about network lifetime is the size of packet headers. If an algorithm performs better in terms of message transmissions and a node uses a protocol with large packet headers its lifetime will increase compared to an algorithm that performs worse in terms of transmitted messages, but probably a bit better in terms of transmitted values when not taking headers into account. Because our algorithm minimizes the overall number of messages, its performance will increase compared to IFILA and EXTOK when using large packet headers.

6. RELATED WORK

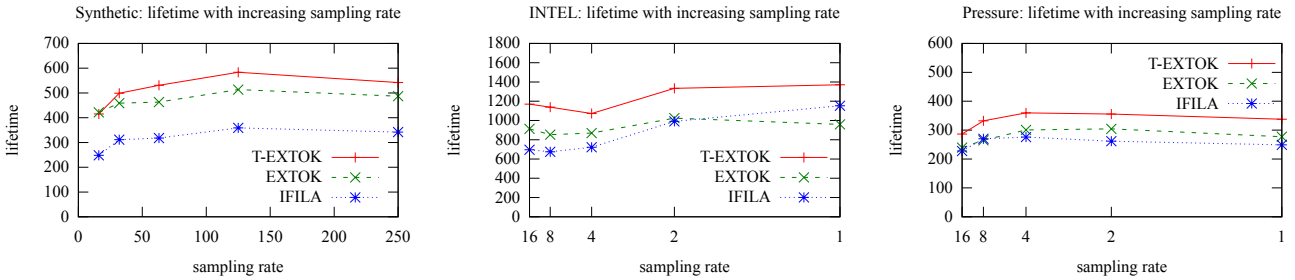

Figure 7: Effect of sampling rate on the network lifetime. Recall that for the synthetic dataset the x-axis denotes the number of samples per cycle, whereas for the real datasets it denotes how many observations exist between gathered sampled observations.

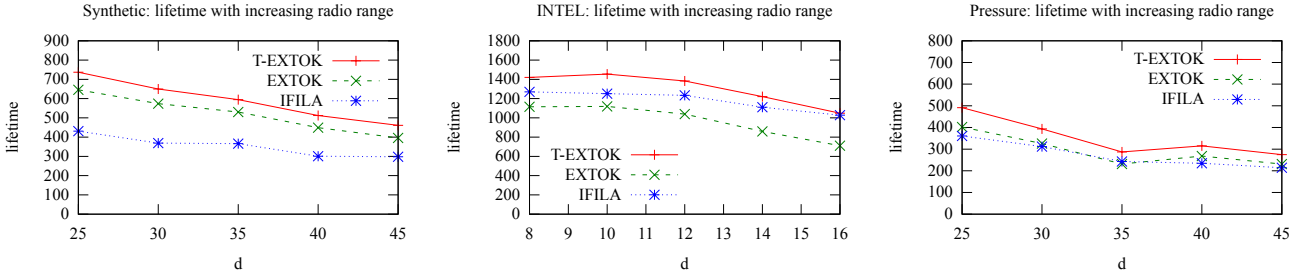

Figure 8: Effect of d on the network lifetime.

TAG [5] is an early solution for performing aggregate queries in wireless sensor networks. In TAG, each routing node that receives more than k values from its children reduces the number of transmitted values to k by removing the smallest values because they will not be relevant for the query at the root node. However, despite reducing the number of transmitted values, the overall number of message transmissions remains relatively large.

Zeinalipour-Yazti et al. [15] suggested the "Threshold Join Algorithm" for performing top-k queries which is applicable for WSNs but not applicable for continuously monitoring the set of top-k values.

Babcock et al. [1] used arithmetic constraints to reduce communication costs for approximate continuous top-k queries that provided a top-k set within specified error bounds. This method of filtering relevant values from all available values has been adapted for other algorithms, e.g., FILA [13]. FILA assigns filters to each node to decide whether or not a node has to send its new measurement, similar to [1]. We refer to Section 2.1 for a more detailed discussion of FILA.

IFILA [11], an improved version of FILA increases the lifetime of a sensor network by computing the filtering intervals in a different manner. It predicts the next measurement of a node using a linear regression model and uses this information to compute the filtering intervals. If two neighboured values show an upward trend, the interval bound is shifted from the midpoint a bit upward, if both show a downward trend the new interval bound is shifted downwards. Their approach of making use of trends is similar to our idea, however, as FILA they transmit different filters to the top-k nodes which increases the number of transmitted values. Besides, if a predicted value increases faster than the distance of a value to its neighbour, IFILA can not take full advantage from trends because filtering intervals are created in a non-overlapping manner where an interval has to contain the most recent measurement. When thinking of fast trends, the new filter that should contain the predicted value of a node would also contain the old value of the neighbouring node, which is not allowed. IFILA also added another improvement that aims on reducing the number of message transmissions while probing. To avoid probing from the root to a node with far distance to the root, it gives routing nodes the ability to probe if necessary.

Although similar to FILA, EXTOK [7] is capable of outperforming FILA and TAG in terms of transmitted values and network lifetime. EXTOK computes the threshold between sets of top-k and non-top-k nodes and broadcasts this separator as a filter. Therefore the set of filters used by FILA can be reduced to a single value. Besides, it explicitly examines ties and is fully topology independent, i.e. the hierarchical topology can change during two consecutive rounds without overhead. This does not only have practical advantages if the tree has to be restructured because a node's energy supply depleted, it can also be used to extend the lifetime of a sensor network by changing the topology and relieve hot-spot nodes. Our solution, T-EXTOK inherits both of these properties.

Other solutions like SLAT, SLAT-A and HAT [10] focus only on MAX-queries, i.e., top-1 queries, and do not examine the case of ties. Besides, HAT is not fully topology independent because routing nodes store thresholds as well.

A novel contribution that examines top-k queries in WSN [4] increases TAG's performance by changing the tree from a shortest path tree to a cluster tree similar to EXTOK. They also suggest another method of utilizing filters for top-k queries. However this filtering approach only makes use

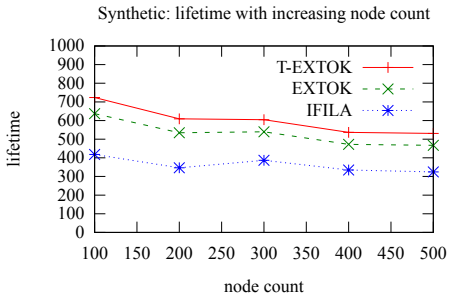

Figure 9: Effect of N on the network lifetime.

of these filters during a round and does not use these filter to predict future measurements as we suggest in this paper.

Another tree topology and querying algorithm was suggested by Cho et al. [2]. They assume that similar sensor measurements are usually locally clustered, for example humidity or temperature, and therefore developed a logical tree that exploits this behaviour. Unfortunately the performance of this algorithm decreases rapidly with increasing spatial uniformity in the underlying data. Although we simulated the behaviour of locally clustered values during our performance evaluation, this assumption is not vital for the performance of our solution.

7. CONCLUSION AND FUTURE RESEARCH

Because of the dependence of wireless nodes on a finite offline power supply, energy saving algorithms for sensor nodes are a very important factor for increasing the usability of this technology. In this paper, we introduced two heuristic approaches for improving the performance of a state-of-the-art algorithm for top-k queries in wireless sensor networks called EXTOK. By using these techniques we were able to get a significant gain compared to EXTOK and IFILA.

In the future we will investigate different use cases where these techniques or even an adapted version of EXTOK is applicable, for example median queries. Median queries are closely related to top-k queries, because computing the $\frac{n}{2}$ top values from a network containing n nodes will return the median of the sensor network. However, using top-k query algorithms for computing medians is still quite inefficient because the root node will have to keep track of $\frac{n}{2}$ nodes.

Acknowledgements

This work was partially supported by Ludwig-Maximilians-Universität (LMU) Munich, Germany, and NSERC Canada.

8. REFERENCES

[1] B. Babcock and C. Olston. Distributed top-k monitoring. In *SIGMOD '03*, pages 28–39, 2003.

[2] Y. Cho, J. Son, and Y. D. Chung. Pot: an efficient top-k monitoring method for spatially correlated sensor readings. In *DMSN '08*, pages 8–13, 2008.

[3] T. Kohonen. *Self-Organizing Maps*. Springer Berlin / Heidelberg, 2001.

[4] X. Liu, J. Xu, and W.-C. Lee. A cross pruning framework for top-k data collection in wireless sensor networks. *MDM '10*, 0:157–166, 2010.

[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[6] A. Mainwaring et al. Wireless sensor networks for habitat monitoring. In *WSNA '02*, pages 88–97, 2002.

[7] B. Malhotra, M. A. Nascimento, and I. Nikolaidis. Exact top-k queries in wireless sensor networks. Technical report, University of Alberta, Edmonton, Canada, http://www.cs.ualberta.ca/research/theses-publications/technical-reports/2009/tr09-16, 2009. To appear at TKDE.

[8] D. Pescovitz. Smart buildings admit their faults. http://coe.berkeley.edu/labnotes/1101smartbuildings.html, 2001.

[9] K. Römer and F. Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11:54–61, 2004.

[10] A. Silberstein, K. Munagala, and J. Yang. Energy-efficient monitoring of extreme values in sensor networks. In *SIGMOD '06*, pages 169–180, 2006.

[11] M. H. Thanh, K. Y. Lee, Y. W. Lee, and M. H. Kim. Processing top-k monitoring queries in wireless sensor networks. In *Sensor Technologies and Applications*, pages 545–552, 2009.

[12] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10:18–25, 2006.

[13] M. Wu, J. Xu, X. Tang, and W.-C. Lee. Monitoring top-k query in wireless sensor networks. *IEEE TKDE*, 19:962–976, 2007.

[14] X. Yan and X. Gang Su. *Linear regression analysis: theory and computing*. World Scientific Co. Pte. Ltd., 2009.

[15] D. Zeinalipour-Yazti et al. The threshold join algorithm for top-k queries in distributed sensor networks. In *DMSN '05*, pages 61–66, 2005.