Towards a Network-aware Middleware for Wireless Sensor Networks

Panayiotis G. Andreou, Demetrios Zeinalipour-Yazti,
George Samaras, Panos K. Chrysanthis[‡]

Department of Computer Science, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus
[‡] Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA
{panic,dzeina,cssamara}@cs.ucy.ac.cy, panos@cs.pitt.edu

ABSTRACT

Wireless Sensor Networks (WSNs), enable users to monitor the physical world at an extremely high fidelity. In order to collect the data generated by these tiny-scale devices, the data management community has proposed the utilization of declarative data acquisition frameworks. While these approaches have facilitated the energy-efficient retrieval of data from the environment, they were agnostic of the underlying network topology, which may impose data reception/transmission inefficiencies. In this paper, we present the architectural design of KSpot$^+$, a distributed middleware framework that introduces *network-awareness* to the data acquisition process by combining three cooperating components: i) the *Tree Balancing Module*, which balances the workload incurred on each sensor node by constructing efficient network topologies; ii) the *Workload Balancing Module*, which minimizes data reception inefficiencies by synchronizing the waking windows of each sensor node; and iii) the *Query Processing Module*, which manages query execution and additionally employs a ranking mechanism that unveils only the k-highest ranked answers thus further minimizing energy consumption.

Keywords

Middleware, Top-K Query Processing, Sensor Networks

1. INTRODUCTION

Technological advances in embedded systems, sensor components and low power wireless communication units have made it feasible to produce small-scale wireless sensor devices that can be utilized for the development of environmental monitoring systems. Large-scale deployments of *Wireless Sensor Networks (WSNs)* have already emerged in environmental and habitant monitoring [23, 20], structural monitoring [13] and urban monitoring [4, 19]. These deployments often employ middleware frameworks that allow users to disseminate queries to the network and collect runtime measurements of sensor data. Since WSNs feature a limited energy budget [16, 20], one of the key design goals of any middleware system is power efficiency.

Our study revealed that although predominant data acquisition frameworks [16, 26, 27, 21, 11, 10, 14, 8, 17, 22] have succeeded in decreasing the overall energy consumption of a WSN by employing power-aware algorithms, they have overlooked the effect of the underlying network topology on data acquisition. In particular, most of the approaches establish query dissemination and data acquisition on the premise of ad hoc *Query Routing Trees (QRTs)*, which provide an effective routing scheme between sensor nodes. However, QRTs do not provide any performance guarantees (e.g., the workload incurred on each sensor node), which leads to significant energy waste [2, 3].

Optimizing the network topology on the other hand, rapidly decreases data reception and transmission inefficiencies. Moreover, additional energy savings can be achieved by closely investigating the query execution process that takes place after generating an efficient topology. Finally, data acquisition middleware frameworks [16, 26, 11, 10, 14, 8, 17, 22] focus only on producing the complete resultset for a query. Conversely, a number of studies [7] model the retrieval of data on the presumption that the user is only interested in the k highest-ranked answers (i.e., `Top-`k queries) rather than all of them. Supporting `Top-`k queries minimizes energy consumption by decreasing both the size and number of transmitted packets in the network by pruning results that will not appear in the final result [28].

In this paper, we present KSpot$^+$, a data-centric distributed middleware architecture that advances the current state-of-the-art by incorporating *network-awareness* and *advanced query semantics* to the data acquisition process for WSNs by combining the following three components:

1. *Tree Balancing Module*, whose objective is to decrease data transmission inefficiencies by transforming the initial QRT into a more balanced QRT in a distributed manner.
2. *Workload Balancing Module*, whose objective is to decrease data reception inefficiencies by automatically tuning the waking window locally at each sensor without any *a priori* knowledge or user intervention.
3. *Query Processing Module*, whose objective is to decrease the number and size of packets transmitted to the network by facilitating advanced query semantics (e.g., `Top-`k, `Group-By` queries).

Figure 1: KSpot$^+$'s prototype implementation deployment during the event "Researcher's Evening" at the Cyprus International Fair in 2009. (Left) Sensor nodes were placed over the pavilions using helium balloons. (Right) Sink station was connected with a laptop computer that projected the pavilions with the highest noise level.

KSpot$^+$ is an open-source middleware framework[1] for WSNs that can be utilized in numerous application domains including environmental monitoring [20], big ephemeral events [25], structural monitoring [13], urban monitoring [19, 4], health monitoring [15], etc. We now describe how KSpot$^+$ has been deployed in the context of two projects:

Environmental Monitoring and Emergency Management: Dynamic monitoring of forests and rivers as well as emergency management require the existence of large sensor network deployments that can provide realtime results. They involve hundreds of sensors and actuators deployed to cover thousand square kilometers of forest areas, thus producing huge amounts of data that require ample time to process. An important factor in sustaining such large sensor network deployments is the cost of maintenance associated with battery replacement. There are solutions today that can rapidly reduce this maintenance cost by utilizing alternative means for energy replenishment including solar panels, bio-harvesting [24], etc. However, in order for these solutions to succeed effectively, the ratio of energy replenishment over energy consumption must be encouraging. KSpot$^+$ decreases energy consumption by minimizing both the size and number of packets, which increases the network's lifespan and reduces maintenance costs.

KSpot$^+$ is currently scheduled to be deployed as part of the FireWatch[2] forest fire monitoring project, sponsored by the Cyprus Research Promotion foundation.

Big Ephemeral Events: International events (e.g., FIFA World Cup, World Expo, etc.) usually attract millions of participants during a very limited period of time. The deployment of smart sensor networks (i.e., sensors, actuators, RFID) in buildings can contribute to improve the visitor's experience by providing new means to easily interact with the surroundings. For example, during these ephemeral big events, affluence-measuring sensors (e.g., sound, proximity) can form logical groups in order to build a compound resource that provides a real-time map of visitors' arrivals at the different pavilions and places and propose visitors an ideal tour, so as to maximize their experience and satisfaction. Additionally, if a crisis situation happens, these compound resources can also help to localize people to rescue. Furthermore, these deployments can be utilized in conjunction with smartphone networks in order to generate opportunistic social networks that form spontaneously according to relationships, which are explicit (e.g., friendship) and/or implicit (e.g., location, energy).

In this context, we have deployed the KSpot$^+$ framework prototype implementation during the event "Researcher's Evening" at the Cyprus International Fair in 2009[3] as shown in Figure 1, and the ICDE'09 [1] conference. Our objective was to create an acoustic map of the pavilions participating in the exhibition and direct the visitors towards the most popular ones (i.e., the most noisy.) This was accomplished by forming logical clusters of the sensor nodes at each pavilion and then measuring the average sound level using the microphone sensor. KSpot$^+$ successfully monitored the pavilions by periodically visualizing the most popular locations (i.e., `Top-3` highest ranked logical groups) every 4 seconds. Additionally, in order to demonstrate the interoperability of the KSpot$^+$ middleware, all acquired results were also recorded in a local database. Noteworthy was that at the end, the organizing committee of the event requested the data trace for further analysis.

This work is founded upon our previous works in [2, 3, 28] where we have presented and evaluated in isolation the three basic components of KSpot$^+$. In this paper, we present an architectural design that enables the seamless cooperation of these components under a unified communication mechanism and additionally argue why such an integration is beneficial.

More specifically, we make the following contributions:

- We present an qualitative comparison of current WSN middleware systems along four dimensions: power efficiency, topology optimization, workload optimization and `Top-`k query support.
- We present a detailed description of the KSpot$^+$ framework's architecture as well as its three basic modules.

The remainder of the paper is organized as follows: Section 2 presents the KSpot$^+$ middleware system architecture and a description of its basic components. Section 3 performs a qualitative comparison of related middleware system research works with KSpot$^+$. Finally, Section 4 concludes our paper.

2. KSPOT$^+$ MIDDLEWARE ARCHITECTURE

KSpot$^+$ is a middleware architecture for wireless sensor networks built on top of energy-conscious algorithms. It inserts a profiling layer between the server and the sensor network that discovers structural and workload inefficiencies and manipulates them in order to generate balanced topologies and query them in an energy-efficient manner. It has three basic operations: i) to construct balanced network topologies; ii) to tune the waking windows of sensor nodes; and iii) to enable tuple ranking through `Top-`k queries.

2.1 Design Goals

In order to build a practical system, we have taken into consideration the following desired properties:

Modularity: Decomposing systems into a number of components that may be mixed and matched in a variety of configurations ensures a high degree of openness and usability of

[1]http://www.cs.ucy.ac.cy/∼panic/kspot/

[2]The FireWatch project, http://firewatch.cs.ucy.ac.cy/

[3]http://crpf.metacanvas.com/EL/int_cooperation/night/index.html

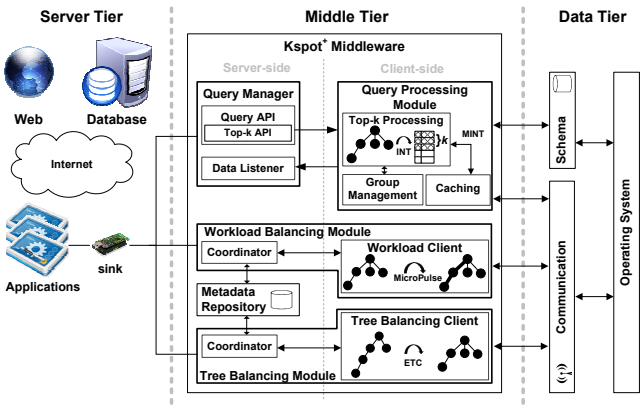

Figure 2: The KSpot$^+$ middleware architecture combines 3 components: the *Tree Balancing Module*, which balances the sensor network topology, the *Workload Balancing Module* , which balances the workload of each sensor node, and the *Query Processing Module* that handles query execution and facilitates Top-k query processing.

our middleware. Our architecture design consists of modular components that eliminate the data transmission/reception inefficiencies and operate in an energy efficient manner both in isolation and in combination with each other as well as with other protocols.

Energy-Efficiency: Battery-powered WSNs are expected to minimize maintenance cost by lasting for large periods of time without requiring battery replacements [16, 28, 23, 3, 2]. To accomplish this, any software (i.e., OS, middleware, application) that runs on a sensor device must be designed and operate in an energy-efficient manner. In the KSpot$^+$ middleware architecture, each module is founded on the premise of energy-conscious algorithms that minimize energy consumption and increase network longevity.

Distributed and Autonomous Behavior: We focus on fully autonomous and decentralized behavior of KSpot$^+$ client-side components. More specifically, we minimize the maintenance of any global state or data structures at a centralized location and use only local knowledge. In the cases where global information is necessary for completing an operation it is acquired using specialized coordinator components. Note, that this only occurs only at the initialization of the network topology upon a balancing request or in case of node failures.

Scalability: The network sizes of WSNs are expected to grow substantially in the next few years as the cost for manufacturing sensor devices continuously decreases. Consequently, we consider scalability an extremely desirable property of our middleware architecture as it ensures that the performance of the system will maintain acceptable QoS standards regardless of the increasing network size.

Accuracy in the presence of Failures: WSNs are typically prone to imminent node failures triggered from temporary power-downs, malfunctions, environmental causes etc. Maintaining accuracy in such environments is vital for applications (e.g., fire detection/prediction) that require exact results. In our previous works [3, 2, 28], we have shown that the algorithms that operate in each of the KSpot$^+$ modules have proven to work accurately and efficiently in the presence of failures.

2.2 KSpot$^+$ Middleware Architecture Design

The KSpot$^+$ middleware lies between the server tier and the data tier as illustrated in Figure 2. Applications can post queries to the sensor network using the Query API or request a balancing operation (Tree Balancing and/or Workload Balancing request) through the respective coordinator components. Queries are forwarded to the Query Processing Module, which in turn decides the best execution plan for the query and communicates with the schema layer in order to retrieve the actual data residing on local storage. As soon as the query results are ready, they are forwarded back to the application through the Data Listener component. Applications can then share the data with online databases and web portals.

Balancing requests require global information, which is stored in the meta-data repository. The Coordinator components recursively forward specialized messages to the sensor network requesting their local values. In the next step, these values are propagated in the opposite order until they reach the sink node. The sink node then calculates the optimal network branching factor (β) and critical path (ψ) values [2, 3] of the network and forwards them back to the coordinator components that proceed with balancing the topology and workload locally at each sensor node.

We now describe in more detail the components of the KSpot$^+$ middleware architecture:

Workload Balancing Module: it investigates data reception/transmission inefficiencies that occur from unbalanced assignment of the query workload amongst sensor nodes. It utilizes the WART algorithm for the dynamic adaptation of the waking windows of each sensor node. The server-side Workload Balancing Coordinator starts by profiling recent data acquisition activity and then identifies the bottlenecks of the network through an in-network execution of the Critical Path Method. The acquired global critical path value (ψ) is stored in the *Meta-data Repository*. Immediately afterwards, the coordinator transmits the ψ value to the network recursively fine-tuning each sensor node's waking window τ locally through the Workload Client.

Tree Balancing Module: it identifies structural inefficiencies in the initial query routing tree that occur from its ad hoc construction nature. It utilizes the Energy-driven Tree Construction (ETC) algorithm in order to remove these inefficiencies by reconstructing the tree in a balanced manner thus minimizing data collisions during communication.

Query Manager: it is responsible for disseminating queries to the network and translating the network results into a tuple-format using the *Data Listener* component. It supports an SQL-like query syntax, which supports standard queries through the *Query API*. Additionally, it extends the SQL syntax of predominant data-centric middleware systems [16, 26, 21, 27] by introducing Top-k query execution in the form of aggregates through the *Top-k Query API*.

Query Processing Module: it is responsible for query execution as well as a number of services including group management and caching. Regular SQL queries are exe-

cuted using the built-in query mechanism while standard Top-k queries are executed using the INT algorithm. In the case of Top-k queries that involve logical groups, these are coordinated through the group management component (see Section 2.5). The Query Processing Module also utilizes a data caching mechanism that in cooperation with the INT mechanism exploits temporal coherency between results of consecutive time instances (MINT).

Data Caching: it exploits the temporal coherency in order to suppress updates that do not change between consecutive time instances. At each epoch, the query results are stored in main memory before they are transmitted so that they can compared with the results of the next epoch. We have chosen to store the results in main memory instead of flash storage because it increases the response time performance of the system.

Group Management: it is responsible for forming clusters of sensors by arranging them in logical groups. This is accomplished by attribute-based naming of the sensors based on specific query semantics and application requirements.

In the following sections, we present selected features of the KSpot$^+$ middleware architecture.

2.3 Modular Design

The KSpot$^+$ middleware architecture is composed of loosely-coupled modules that communicate with message-passing. As illustrated in Figure 2, the server-side components communicate with their client-side *accomplices* using different communication messages (different arrows departing from each server-side component.) The reason we have not opted for a unified communication mechanism is that this generates a tightly-coupled system, which would have compromised the modularity of the system as tightly coupled systems tend to exhibit a number of disadvantages including: i) *Decreased Reusability*, because dependent modules must be bundled together in order to be reused or tested; ii) *Increased Deployment Effort*, because module bundles will require more time to test and deploy; and iii) *Increased Maintenance*, as updates on one module may require re-testing of the whole bundle.

KSpot$^+$ modular design allows application designers to easily integrate new features into the design as well as to experiment under different settings. KSpot$^+$'s modules can function individually or in cooperation according to the requirements of the application. Additionally, they can operate in conjunction other routing and query protocols.

2.4 Query Syntax

The KSpot$^+$ supports an SQL-like query syntax similar to [16, 26, 21, 27], which supports standard queries through the *Query API* and Top-k queries through the *Top-k API*.

In particular, the KSpot$^+$ middleware architecture utilizes the following query syntax:

```
SELECT Top k attribute [,aggregate]
FROM sensors
[WHERE filter]
[GROUP BY attribute]
[ORDER BY [attribute|aggregate] [ASC|DESC]]
[SAMPLE PERIOD time (ms)]
```

The *attribute* parameter mentioned in the SELECT statement refers to all measurements that can be acquired from the sensorboard as well as variables stored locally at each sensor node. The *attribute* parameter mentioned in the GROUP BY statement may additionally refer to a logical group assignment, as described in the next Section. The *aggregate* parameter refers to all aggregates supported. Roughly, these aggregates can be distinguished in: i) *distributive aggregates*, where records can be aggregated in-network without compromising correctness (e.g., MAX, MIN, SUM), and ii) *holistic aggregates*, where in-network aggregation might compromise the result correctness (e.g., MEDIAN), thus all tuples have to be transmitted to the sink before the query can be executed.

Note that in the KSpot$^+$ framework, when a Top-k attribute query is executed over the network, we only return the k-highest results for that attribute, if no ORDER BY clause is used. However, we could have easily returned the k-lowest results in a similar way. If a GROUP BY query is posted to the network, results are grouped by the attribute statement and aggregates are calculated for each group individually.

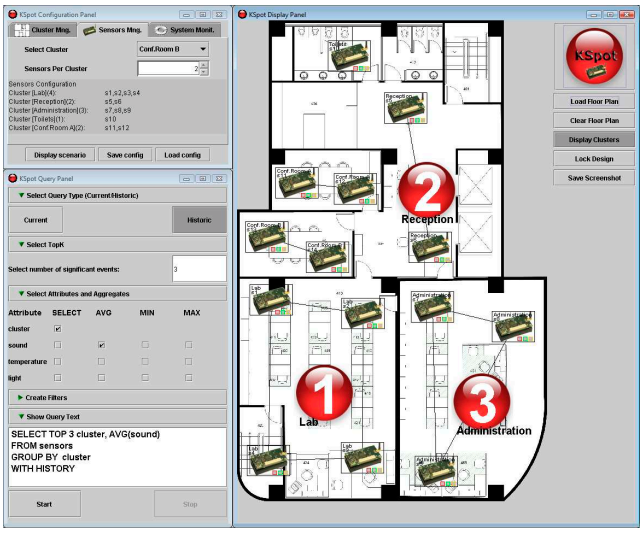

Figure 3: KSpot$^+$'s Graphical User Interface (GUI) allows users to administer the execution of standard and Top-k Queries through an intuitive and declarative graphical user interface. The above scenario conducts a Top-3 query over a 14-node sensor network organized in 6 logical clusters. The Display Panel (on the right) illustrates the three KSpot$^+$-Bullets for the three highest-ranked sensor clusters.

2.5 Logical Group Management

The Group Management component realizes clustering of the sensor nodes by arranging them into logical groups. This is necessary in the case of GROUP BY queries, where grouping may be achieved not only on predefined attributes (e.g., nodeid) but also on context-based attributes (e.g., building name, room number). To facilitate our description, consider an indoor deployment of four sensor nodes s_{1-4} in a building with two offices, A and B, such that s_{1-2} is located in office A and s_{3-4} in B. In order to inform each sensor node on its actual location (e.g., longitude, latitude) and then derive its logical location (i.e., office A or B), we could have utilized absolute localization techniques (e.g., Global Positioning System (GPS)) or relative localization techniques (e.g., RSSI indicators) and then perform the logical map-

ping on the server. However, this requires specialized hardware (e.g., GPS receiver, beacons), which may not be always available in indoor environments and also increases the overall message complexity. To overcome this, the KSpot$^+$ API, supports commands for creating and deleting logical groups that are injected to the network and processed locally at the sensor nodes.

2.6 Proof of Concept Application

In order to assess the practicality and usability of the proposed KSpot$^+$ framework, we have developed a proof of concept application (KSpot$^+$ POCA) that demonstrates the full potential of KSpot$^+$ (see Figure 3). KSpot$^+$ POCA components are implemented in JAVA (server-side) and in nesC (client-side). We have selected nesC for the implementation of the client-side components for practical reasons as it provides a kernel of declarative data acquisition functionalities (i.e., SQL query syntax). However, we could have similarly applied our ideas on other sensor network operating systems (e.g., LiteOS [5]).

3. RELATED WORK

Traditional middleware systems such as the Common Object Request Broker Architecture (CORBA) [6], Java service oriented architecture (JINI) [12], Enterprise Java Beans (EJB) [18] are considered heavyweight in terms of processor and memory requirements thus are highly inefficient for WSN deployments. In order to tackle this problem many research works have proposed lightweight and energy efficient middleware architectures tailored specifically for WSNs.

Cougar[26] and TinyDB[16] are two typical data-centric middleware systems closely related to KSpot$^+$, which view the network as a virtual relational database and inject query messages to the network that are then processed locally at each sensor node. Similar to KSpot$^+$, Cougar [26] employs a centralized optimizer, which maintains status information about the network in order to coordinate sensor nodes in an energy-efficient manner. However, in [2] we have shown that node and communication failures severely hamper the efficiency of this coordination scheme as they cause sensor nodes, especially the ones in higher levels, to stay in reception mode longer than required. TinyDB [16] is a more mature data acquisition middleware system supporting SQL-like queries of various types (e.g., event-based, lifetime, etc.) and in-network aggregation. TinyDB's power-aware optimizer employs a cost-based mechanism in order to choose the most energy-efficient query execution plan. This often enforces a uniform waking window for all sensor nodes depending on the depth of the QRT, which in the majority of the cases it is clearly an overestimate. The rationale behind this over-estimation is to offset the limitations in the quality of the underlying clock synchronization algorithms of the operating system but in reality it is too coarse [2].

The Sensor NEtwork Engine (SNEE) [9] also employs an optimizer that receives metadata information about the available resources (e.g., memory, energy), the WSN topology and also predictive cost models. SNEE combines a rich, expressive query language, named SNEEQL, which provides extensive support for JOIN operators incorporating techniques found on classical DQP architectures. Unlike KSpot$^+$, the proposed query language does not directly address `Top-`k queries although we assume that they can be incorporated as an aggregate function. Furthermore, SNEE supports workload balancing by scheduling different workloads to different sites in the network thus effectively reducing the energy. However, SNEE assumes that the underlying infrastructure employs an efficient protocol for self-organization of the topology thus neglecting to investigate the effects of an unbalanced topology. KSpot$^+$ addresses the latter with the aid of the Tree Balancing Module. Sensor Information Networking Architecture (SINA) [21], provides a set of configuration/communication primitives that enable scalable and energy-efficient organization as well as query-processing. Additionally, a cluster-based protocol achieves minimization of data transmission/reception inefficiencies by sacrificing the results of some sensors. This however, may result in generating inaccurate results at the sink node. (DsWare) [27] employs a filtering mechanism that provides approximate instead of exact values in order to decrease communication overhead. In KSpot$^+$, we minimize the overall energy consumption of the network while in parallel maintaining high degrees of accuracy.

Table 1: Classification and Comparison of Middleware Approaches for WSNs

Middleware Approach	Energy-aware	Workload Opt.	Topology Opt.	Top-k Queries
Data-centric				
TinyDB [16]	Y	Y	N	N
Cougar [26]	Y	Y	N	N
SNEE [9]	Y	Y	N	N
DsWare [27]	Y	N	N	N
SINA [21]	Y	N	N	N
KSpot$^+$	Y	Y	Y	Y
Application-driven				
Milan [10]	Y	N	Y	N
MidFusion [11]	Y	N	N	N
Publish-Subscribe				
Mires [22]	Y	N	N	N
AWARE [17]	Y	Y	N	N
Agent-based				
Impala [14]	Y	Y	N	N
Agilla [8]	Y	Y	N	N

Other middleware approaches include application-centric [10, 11], publish-subscribe [22, 17] and agent-based [14, 8] middleware systems. Application-centric approaches like the Middleware Linking Applications and Networks (Milan) [10] and MidFusion [11] allow application designers to specify their QoS requirements inside the sensor network application code. In Milan, these requirements are then translated by the middleware into protocol-specific commands that configure and manage the network according to heuristics similarly to KSpot$^+$. Unlike KSpot$^+$, Milan does not consider the workload incurred on each sensor node, which may result in serious data reception inefficiencies. MidFusion discovers and selects the best set of sensor nodes (w.r.t. energy efficiency) that can respond to an application request. This process may omit sensor nodes from the data acquisition process because of the application QoS requirements, which may lead to inaccurate results. Publish-subscribe middleware systems [22, 17] allow each device to publish its capabilities (i.e., data channels) and attributes, to a centralized registry where other devices can subscribe to and receive feeds. Although, this setting can support a number of packet-level optimizations that can greatly decrease the number of communication packets, additional energy savings can be achieved by optimizing the network topology. Agent-based middleware systems [14, 8] employ agent-based components that enable dynamic adaptation of running applications in order to improve performance, energy-

efficiency and robustness. This ensures a high degree of energy-efficiency without user-intervention. On the other hand, the coordination of mobile agents on a large-scale network may seriously hamper the performance of the network.

In conclusion, all proposed middleware systems employ mechanisms for reducing the overall energy consumption of the network thus increasing the longevity of a WSN. However, they neglect the important parameter of constructing an energy efficient topology [3] and operate on top of the initial ad hoc query routing tree. Additionally, most approaches often assume a fixed workload distributed uniformly on all sensor nodes. Consequently, it is not clear how efficient they will operate under a variable workload [2], which occurs under the following circumstances: i) from a non-balanced topology, where some nodes have many children and thus require more time to collect the results from their dependents; and ii) from multi-tuple answers, which are generated because some nodes return more tuples than other nodes (e.g., because of the query predicate). Finally, none of the approaches support `Top-`k queries, which can significantly decrease the overall number and size of transmitted packets [28]. Like all presented approaches, the KSpot$^+$ middleware architecture focuses on energy efficiency but additionally employs mechanisms that generate a more efficient topology as well as provide support for `Top-`k queries. Table 1, summarizes the results of our analysis.

4. CONCLUSIONS

Current middleware approaches for WSNs may suffer from data reception/transmission inefficiencies because they operate on the presumption that the underlying network topology is efficient. In this paper, we have presented KSpot$^+$, an energy-efficient data-centric middleware architecture that provides mechanisms for measuring and improving the efficiency of the network topology in order to decrease the cost of data acquisition. Additionally, KSpot$^+$ employs a ranking mechanism that unveils only the k-highest ranked answers thus further minimizing energy consumption.

5. ACKNOWLEDGMENTS

This work was supported in part the second author's Start-up Grant, funded by the University of Cyprus, the Open University of Cyprus under project SenseView, the US National Science Foundation under projects S-CITI (#ANI-0325353) and AQSIOS (#IIS-0534531), the European Union under the projects IPAC (#224395) and CONET (#224053), and the project FireWatch (#0609-BIE/09), sponsored by the Cyprus Research Promotion foundation.

6. REFERENCES

[1] P. Andreou, D. Zeinalipour-Yazti, M. Vassiliadou, P. K. Chrysanthis, G. Samaras, "KSpot: Effectively Monitoring the K Most Important Events in a Wireless Sensor Network", In ICDE'09.

[2] P. Andreou, D. Zeinalipour-Yazti, P. Chrysanthis, G. Samaras, "Workload-aware Query Routing Trees in Wireless Sensor Networks", In MDM'08.

[3] P. Andreou, A. Pamboris, D. Zeinalipour-Yazti, P.K. Chrysanthis, G. Samaras, "ETC: Energy-driven Tree Construction in Wireless Sensor Networks", In SENTIE'09.

[4] A.T. Campbell, S.B. Eisenman, N.D. Lane, E.Miluzzo, R.A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, G-S. Ahn, "The Rise of People-Centric Sensing", In IEEE Internet Computing'08.

[5] Q. Cao, T. Abdelzaher, J. Stankovic, T. He, "The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks", In IPSN'08.

[6] Common Object Request Broker Architecture (CORBA), http://www.corba.org/

[7] R. Fagin, "Combining Fuzzy Information from Multiple Systems", In JCSS'99.

[8] C-L. Fok, G-C. Roman, C. Lu, "Agilla: A mobile agent middleware for self-adaptive wireless sensor networks", In ACM TAAS'09.

[9] I. Galpin, C.Y.A. Brenninkmeijer, F. Jabeen, A.A.A. Fernandes, N.W. Paton., "An Architecture for Query Optimization in Sensor Networks", In IEEE ICDE'08.

[10] W.B. Heinzelman, A.L. Murphy, H.S. Carvalho, M.A. Perillo, "Middleware to support sensor network applications", In IEEE Network'04.

[11] A. Hitha, K. Mohan, S. Behrooz, "MidFusion: An adaptive middleware for information fusion in sensor network applications", In Information Fusion'08.

[12] Java service oriented architecture (JINI), www.jini.org/

[13] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, M. Turon, "Health Monitoring of Civil Infrastructures using Wireless Sensor Networks", In ACM IPSN'07,

[14] T. Liu, M. Martonosi, "Impala: a middleware system for managing autonomic, parallel sensor systems", In SIGPLAN Notices'03.

[15] K. Lorincz, B. Chen, G.W. Challen, A.R. Chowdhury, S. Patel, P. Bonato, M. Welsh, "Mercury: A Wearable Sensor Network Platform for High-fidelity Motion Analysis", In ACM SenSys'09.

[16] S.R. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, "The Design of an Acquisitional Query Processor for Sensor Networks", In ACM SIGMOD'03.

[17] A. Ollero, M. Bernard, M.L. Civita, L. van Hoesel, P.J. Marron, J. Lepley, E. de Andres, "AWARE: Platform for Autonomous self-deploying and operation of Wireless sensor-actuator networks cooperating with unmanned AeRial vehiclEs", In IEEE SSRR'07.

[18] Oracle Enterprise JavaBeans Technology (EJB), http://www.oracle.com/technetwork/java/javaee/ejb/index.html

[19] I.Rose, M. Welsh, "Mapping the Urban Wireless Landscape with Argos", In ACM SenSys'10.

[20] C. Sadler, P. Zhang, M. Martonosi, S. Lyon, "Hardware Design Experiences in ZebraNet", In ACM SenSys'04.

[21] C-C. Shen, C. Srisathapornphat C., C. Jaikaeo, "ensor information networking architecture and applications", In IEEE Personal Communications'01.

[22] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, J. Kelner "Mires: a publish/subscribe middleware for sensor networks", In Personal Ubiquitous Computing'05.

[23] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, D. Culler, "An Analysis of a Large Scale Habitat Monitoring Application", In ACM SenSys'04.

[24] Voltree Power Inc., http://www.voltreepower.com/

[25] D.B. Yang, H.H. Gonzalez-Ba, L.J. Guibas, "Counting People in Crowds with a Real-Time Network of Simple Image Sensors", In IEEE ICCV'03.

[26] Y. Yao, J.E. Gehrke, "The cougar approach to in-network query processing in sensor networks", In SIGMOD'02.

[27] X. Yu, K. Niyogi, S. Mehrotra, N. Venkatasubramanian, "Adaptive Middleware for Distributed Sensor Environments", In IEEE DS Online'03.

[28] D. Zeinalipour-Yazti, P. Andreou, P.K. Chrysanthis, G. Samaras, "MINT Views: Materialized In Network Top-k Views in Sensor Networks", In MDM'07.