Energy Efficient Query Processing for WSN based on Data Caching and Query Containment

Kayiram Kavitha
Lecturer
Dept. of CS & IS
BITS-Pilani, Hyderabad Campus, Hyderabad, India

Vinod Pachipulusu
Decision Scientist
MuSigma Business Solutions, Bangalore, India

Sreeja Thummala
Student
Dept. of CS & IS
BITS-Pilani, Hyderabad Campus, Hyderabad, India

R.Gururaj, Ph.D
HOD & Asst. Prof.
Dept. of CS & IS
BITS-Pilani, Hyderabad Campus, Hyderabad, India

ABSTRACT

Wireless Sensor Networks (WSNs) are deployed to capture the sensed data from tiny sensors spread around the physical environment. In general, WSNs are used to monitor physical phenomena like temperature, pressure, humidity etc. In most of the cases they are deployed in remote geographic locations and operate unmanned. Usually, these sensors are battery operated. Due to these deployment circumstances, battery recharge or replacement becomes almost impossible. Hence, the foremost requirement of any WSN is to utilize the battery power in an efficient way. A sensor node expends most of its energy in data transmission. It is observed that a query submitted to WSN may request same data or subset of data as that of another request. In this paper, a novel query processing scheme is proposed that exploits the cached results at the BS and the commonality among the queries which require data from the network. This can significantly minimize the transmission and processing costs w.r.t., energy in the network. The experimental results proved the same.

1. INTRODUCTION

A Wireless Sensor Network (WSN) [1] is a collection of multiple sensor nodes connected together to form a network. The applications of WSN include Wastewater Monitoring, Green House Monitoring, Air Pollution Monitoring, Machine Health Monitoring, Landslide Detection, Forest Fire Detection etc. These applications monitor certain physical characteristics like temperature, pressure, humidity, light, air pollutant etc. A sensor node is a tiny electronic device with a small micro controller, memory unit, energy source (usually battery) and a radio transceiver for wireless communication. The role of each sensor node is to sense and transmit the sensed data to the central monitoring station termed as the Base-Station (BS). It is assumed that the BS has abundant computational, storage and energy resources. The battery attached to the sensor node is the source of energy for performing various operations like sensing, data storage, computations and transmission. It is observed that 80% of the power is utilized towards radio communication. Hence, to increase the lifetime of the network, it is essential to minimize the volume of data transmission during query processing.

As the BS links the WSN with the outside world, it is assumed that all queries are submitted to the BS. In general, the queries submitted by end-users/applications may be either ad-hoc [2] or continuous [3]. In case of ad-hoc queries, it requires one time execution only. For example, “Give the maximum temperature today” is an ad-hoc query. On the other hand continuous queries refer to such requests which require processing at some specified intervals of time or on some event. For example, the query “Give the average temperature for the last 10 hours on hourly basis” is a continuous query.

In this work, it is assumed that both BS and the sensor nodes are capable of processing queries. The only difference between a sensor node and BS is in terms of the resources available. Further, in this proposal the focus is on minimizing the cost of executing ad-hoc queries only. Usually, an ad-hoc query submitted at the BS is sent to respective network of nodes for execution. Since, the results of a query are given back to the user through the BS; all the nodes send their result data to the BS. As the BS has a reasonable amount of storage, the results of a query can be stored at the BS for future use. Such data stored at the BS is called as cached result. If the query request involves cached results, the BS can process the query and give the results. Otherwise, the query needs to be sent to the concerned nodes for execution.

The earlier work in [4] optimizes query execution by identifying queries which can be answered at BS based on the domain knowledge that defines certain constraints on the data values. But by using domain knowledge, BS can answer only certain queries without disseminating the query into the network. The approach used in [5] exploits spatial containment among queries. According to this, queries request data from a particular geographical region specified as bounding rectangle co-ordinates. The bounding rectangle co-ordinates of each query are mapped onto R-tree for identifying the largest bounding rectangle, based on which a single query is formulated that covers all the required regions, and executed in the network. This avoids dissemination and execution of queries representing the bounded rectangles contained by the largest bounded rectangle. But, this approach does not exploit attribute level and predicate level commonalities. Though the techniques mentioned in [6] [7] try to exploit the query containment, they look at the containment w.r.t., result attributes only. They have not considered the filters specified in the form of conditions in the WHERE clause of the query. All these optimization techniques lead to energy savings in specific application contexts. Hence, the techniques are highly application dependent. Full length detailing about the above approaches is given in Section 2 as part of the related work.

In this work, the focus is on set of ad-hoc queries received at BS, which need to be executed at BS and/or at network nodes. A set of ad-hoc queries submitted at BS are often found to be requesting the same results. Sometimes they may not be equal, but they exhibit certain overlap in their query results. Such relationship among queries is termed as query containment. The containment relationship is said to be satisfied, if two queries yield same or partially same query results. For the sake of convenience w.r.t., terminology, a super-query can be defined as a query ‘q’ that yields the superset of the results of a set of queries Q {q1….qn}. This super-query ‘q’ is formed based on ‘Q’. Hence, the challenge is to formulate a super-query-set for a set of queries which can be disseminated and executed in the network leading to reduced transmission cost in terms of energy. Further, a query submitted at BS may need to be processed- (i) completely using the cached data at BS or (ii) completely against the data available in the network, or sometimes (iii) partly against the cached data or partly against the data available in the network. In this paper, a novel query processing technique is proposed that exploits cached results at BS and query containment among the ad-hoc queries to be executed in the network. If the query results can be obtained in its entirety from the cached results then the query can be executed against the cached results at the BS and send the results to the user. Otherwise if the query results need to be extracted partially or completely from the network nodes, then the commonalities w.r.t., the data requirements of all such ad-hoc queries need to be found to formulate super-queries and transmit the same to the required nodes for execution. In this work, a bit-map approach is proposed for identifying the commonalities (containment) among the queries and formulating a super-query. This helps in minimizing the number of queries executed and the volume of data transmitted in the network which eventually leads to optimal utilization of the power and longer network life. The experimental results prove the effectiveness of this approach for WSNs requiring to process ad-hoc in-network queries.

The rest of the paper is organized as follows. The related work and limitations are presented in Section 2, and the proposed solution is described in Section 3. Section 4 shows the simulation results and analysis. Finally, the paper concludes in Section 5.

2. RELATED WORK

The work in [4] proposes an approach for processing aggregate queries with the help of domain knowledge and sensing constraints. An aggregate query may involve one or more of the operations like- average, sum, minimum, maximum, and count. For instance, consider the query “Is the average of the temperatures of the nodes n_1 and n_2, greater than 80”. To process this query, the BS collects temperature from n_1 and n_2 and computes the average and checks if it is greater than 80. The result of this query is Boolean. If the domain semantics specifies that the temperature at any node is between 0 and 100, this knowledge can be used by BS in simplifying the execution. For example, to process this, BS can retrieve one of the temperatures first and check if it is less than 60. If so, it tells that independent of the temperature reading at the other node the result is false. In that case, it need not send and process the query for n2. This way unnecessary query/result transmission and processing are avoided. This approach can be applied only for certain queries, which use universally fixed knowledge and constraints.

The Query model in [5] allows queries to specify geographical area rather than explicitly stating single data sources. Each query region is specified as a bounding rectangle using co-ordinates. Two queries q1 and q2 are said to exhibit containment relation, when the query, q1 is found to request data from the sensors within the geographical boundaries specified by q2 or vice-versa. This is identified by constructing R-tree, which stores the bounding rectangle co-ordinates for each query. This R-tree is used to identify the largest rectangle that contains all other rectangles. Now, the BS will disseminate queries to the respective nodes contained in the rectangle. The individual nodes process the queries and transmit the results back to the BS where it is recompiled to answer individual queries. This scheme eliminates redundancy in query and result transmission, and significantly increases the query efficiency. The drawbacks in R-tree approach are as follows. While constructing R-tree, the containment among the queries is identified by looking at the geographical co-ordinates specified in individual queries. Hence this containment relationship does not take the conditions specified in the WHERE clause of the query i.e., filters applied on other parameters like time etc. This is the major drawback of the R-Tree algorithm. As it is observed, that in most of the wireless applications data retrieval mainly based on the conditions specified in WHERE clause with respect to time, it is believed that if the containment relationship is characterized by time points, it is more likely to increase the effectiveness.

The work in [7] presents formulation of the problem of query scheduling at the BS, and also solves this with an efficient query scheduling algorithm. The paper solves the problem of scheduling queries with different sliding window sizes and different frequency upper bounds to share computation. The problem is formulated with these two different variations over data streams, and also proposes a combination rule to classify queries. The novel scheduling algorithm is to share the computation between similar queries. Two queries are compared for identifying their common tasks. This comparison is done based on the frequency semantics of the query. The scheduling algorithm uses Earliest Deadline First (EDF) to solve the situations of overload and under-load of the system. This approach is limited to the semantics of frequency and different sliding window sizes in a query. This scheduling scheme does not consider event detection queries and continuous queries.

With advancements in mobile devices, the sensor networks are made available to mobile users wherever they arrive at. A single sensor network supports multiple applications for mobile users. The work in [8] deals with merging the queries coming from different devices. These users can query the WSN in different query formats and with different query clauses. But, all the queries request data from the same sensor network. So, two or more users may request same data or partially same data. Therefore, it is highly desirable to optimize the query set before disseminating into the network. The work in [8] deals with merging the incoming queries from various sources and writing a network query which is sent into the network. The attributes from the user queries form attribute list in the network query. The sampling period in the network query is computed as the least sampling period among the user queries. The network query formed will be a single giant query disseminated into the network. The results obtained from the network are filtered into their respective query results. The query merging does not consider aggregate queries and queries with GROUP BY clause and HAVING clause and hence limits the scope for optimization.

To the best of our knowledge, none of the works done so-far exploited the WHERE clause predicates of a query. Therefore, a novel scheme is proposed to improve the energy efficiency, by exploiting the cached results at BS and the containment relationship among the queries submitted at the BS. According to this, super-queries are formulated that contain the results of all the queries related by containment.

3. PROPOSED SOLUTION

In this Section a detailed account of the proposed query processing scheme for WSNs is given.

3.1 Overview of workflow for query processing in the proposed solution

The external world submits queries at the BS. The queries submitted at the BS are stored in a query register. Now, each query from the query register may obtain its results completely or partially on the cached results at BS. Sometimes, the cached results may not be useful in any way. In case a query finds its complete results at the BS, then the query need not be disseminated into the network. But, if a query yields partial results at the BS then the remaining results are obtained from the network of nodes. For certain queries which don't find results at the BS, need to get their query results from the network. When examined the queries which need data from the network may exhibit certain overlap in their query results. For a given set of queries, that are submitted at BS in a given time interval the data requirements using a bit-map is captured. After elapse of the time period, a set of super-queries are framed from the bit-map which will be disseminated into the network. The query results obtained from the network of nodes will be stored in cached results at the BS. Now, the BS has complete results for all the queries in the register. Therefore, the queries from the query register are executed upon the cached results. The query results are given to the user.

3.2 Query Processing based on the cached results at BS

With the available cached results at the BS, each query in the query register may yield complete, partial or no results. But, the challenge is to identify the data requirements of each query. To solve this problem, a simple technique is proposed that uses a bit-map. In general, a WSN is queried for the sensor value(s) during a fixed time interval. The query results are with two columns viz., the time-stamp and its sensor values. Now, an empty two column query result table is framed, where the first column values are generated with the time-stamp for the required time interval. The second column is the sensor values for their respective time-stamps, which can either be obtained from the cached results at the BS or from the network of nodes. Therefore, sensor values are fetched for the required time-stamp from the cached results at the BS if available. For the sensor values which are not available in the cached results they need to be retrieved from the network of nodes. Thus, values for the second column are available either for all rows or only for some rows. The following situations arise based on the values in the second column of the query result table-

1. The second column is completely filled that means query results are available in the cached results at the BS.
2. The data is partly filled, that means there are empty locations for which data is unavailable at the BS.
3. The second column is empty, which means that the query cannot be answered at the BS.

In the first case, the query can be completely answered at the BS itself without transmitting the query into the network. The second case gives partial results at BS and the rest from the network. The third case is to completely retrieve from the network. In the second and third cases, where the data needs to be retrieved from the network, there may be certain redundant data requirement among the queries. Hence, it is desirable to minimize such redundant data transmissions in the network. Therefore, a simple technique to solve this is using the bit-map approach, presented in Section 3.3. The flowchart for the same is shown in Fig 1.

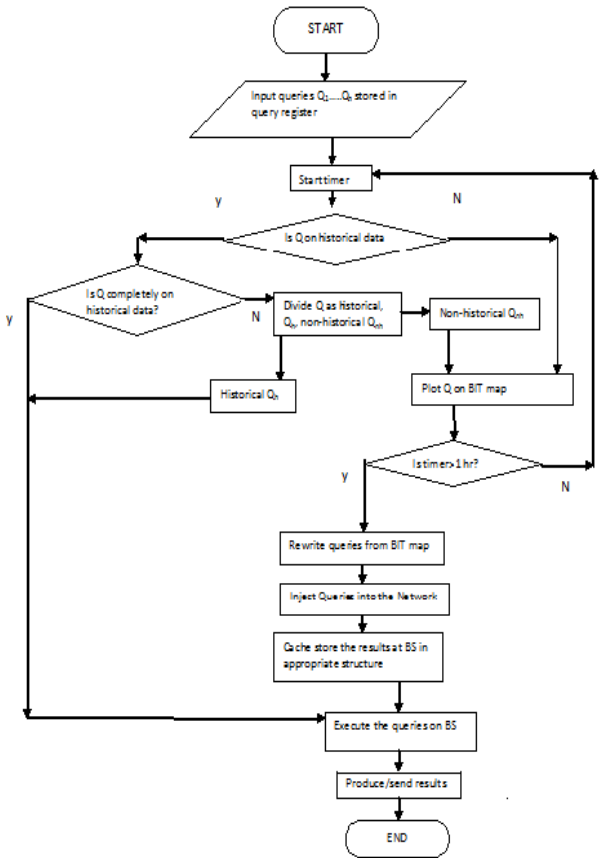
START
Input queries Q_1.....Q_n stored in query register
Start timer
y
N
Is Q on historical data
Is Q completely on historical data?
y
N
Divide Q as historical, Q_h, non-historical Q_{nh}
Non-historical Q_{nh}
Historical Q_h
Plot Q on BIT map
Is timer>1 hr?
y
N
Rewrite queries from BIT map
Inject Queries into the Network
Cache store the results at BS in appropriate structure
Execute the queries on BS
Produce/send results
END

Fig 1: Flowchart showing the proposed solution

3.3 Identifying query containment using bit-map

In general, the queries which do not yield results at the BS are disseminated into the network. The data requirements of these queries are captured in a 2D bit-map. Now, the details of the bit-map are explained. A bit-map is an array of bits for each sensor. Each bit position in the bit-map represents a time-interval. The queries obtained from second and third case in Section 3.1 are the candidates for the bit-map. The commonalities in multiple query results are exploited by mapping all the query requirements onto the bit-map. Initially, the bit-map has '0' for all its bits. The data required at a particular time interval is marked as bit '1' against their time-interval. Whenever bit '1' is found existing in a cell, ignore that and proceed. It means this particular data has already been in requirement of some other query. Now, all the incoming queries are plotted on the same bit-map for a certain time period. Depending on the application, the time period can be set. After the time period is elapsed, a set of super-queries are framed from the bit-map, which is explained in Section 3.4.

3.4 Framing super-query from the bit-map

At the BS, super-query-set from the bit-map are framed on hourly basis or once every six hours depending on the application and user requirement. With an increase in this time period, the number of queries plotted on bit-map also increase. To frame the super-query-set from the bit-map, scan each bit-map from left to right.Whenever there is an entry '1' , its corresponding time-stamp is considered as the lower bound and continue scanning its adjacent cell for entry '0' to indicate the upper bound for the time series. Now, the upper and lower bound time stamps are obtained for each sensor from the bit-map. This forms the data requirement for each sensor. Hence, super-query for each sensor with these unique time-stamps can be framed. The same process is repeated for all bit-maps to obtain a set of super-queries. Now, the bit-maps are set to '0' and repeat the proposed query processing at the BS.

3.5 Dissemination of super-query and result compilation

The super-queries framed in Section 3.4 are disseminated into the network. The query results obtained from the network are cached at the BS. Each query in the query register is now executed upon the cached results at BS. Hence, the query results are delivered to the user. Now, an illustration for the proposed approach is presented in Section 3.6.

3.6 Illustration for the proposed approach

Here, using an example, the working of the proposed approach is illustrated. First, the initial query processing at the BS is explained, and then identification of query containment using bit-map approach is detailed. For simplicity, a network with 4 sensor nodes is considered and time interval for each sensor node is assumed as one hour. The 5 queries {q_1, q_2, q_3, q_4, q_5} are as follows.

q_1: SELECT * FROM S1, S2 WHERE t > 1 AND t < 2;
q_2: SELECT * FROM S1, S3 WHERE t > 2 AND t < 3;
q_3: SELECT * FROM S1, S4 WHERE t > 3 AND t< 5;
q_4: SELECT * FROM S3, S2 WHERE t > 1 AND t< 4;
q_5: SELECT * FROM S4, S2 WHERE t > 3 AND t< 6;

A sample bit-map is shown in Fig 2 below. A bit-map is meant to capture the data requirements for each sensor. The data required from each sensor for a particular time interval is shown as a bit-map. Initially the bit-map is empty, which is shown with '0' in all its bit-positions.

Sensor nodes/Time	1	2	3	4	5	6
S1	0	0	0	0	0	0
S2	0	0	0	0	0	0
S3	0	0	0	0	0	0
S4	0	0	0	0	0	0

Fig 2:Sample bit-maps showing initial value '0' in all cells.

Let us assume that initially the BS has data from Sensor nodes S1, S2, S3, S4 for time interval, t=1 to t=2, which forms the cached results. The query q_1 needs data from Sensor nodes S1, S2 for time interval t=1 to t=2, which is available at the BS. Therefore, q_1 is answered at BS itself. Now q_2 needs data from Sensor nodes S1, S3 for time interval t=2 to t=3 which is not available at the BS. So, the query is plotted onto the bit-maps of Fig 2. Now, the value 1 is plotted in the bit-maps of S1, S3 against the time interval t=2 to t=3 and the modified bit-maps as shown in Fig 3 is obtained.

Sensor nodes/Time	1	2	3	4	5	6
S1	0	1	1	0	0	0
S2	0	0	0	0	0	0
S3	0	1	1	0	0	0
S4	0	0	0	0	0	0

Fig 3: Bit-maps after plotting q_2.

Now consider query q_3, which needs data from S1, S4 for a time-interval t=3 to t=5. This data is not available at the BS. So, plot this on the bit-map obtained in Fig 3. Let us take up the query q_4, which needs data from S3, S2 for a time interval t=1 to t=4. First it is executed on cached data at the BS. The data is in cache at BS for a time interval of t=1 to t=2. But, data from t=2 to t=4 is needed. Hence this unavailable data is plotted on the bit-maps. In case if a particular bit position in the bit-map has value 1 before plotting, this can be ignored, as it has already been the requirement of previous query. Similarly for query q_5, the bit-maps obtained after the previous query q_4 will be the input bit-map. The Final bit-map after plotting all 5 queries is shown in Fig 4 below.

Sensor nodes/Time	1	2	3	4	5	6
S1	0	1	1	1	1	0
S2	0	1	1	1	1	1
S3	0	1	1	1	0	0
S4	0	0	1	1	1	1

Fig 4: Bit-map after plotting q_2, q_3, q_4, and q_5.

Now, the Fig 4 is considered for generating super-queries. Each row specifies the required data from each sensor for their time interval. Like Sensor S1 needs data from the network for a time interval of t=2 to t=5. Hence the super-query that can be generated is SELECT * from S1 where t > 2 and t < 5. Similarly, super-queries for the other sensors can be given as -

SELECT * from S2 where t > 2 and t < 6.
SELECT * from S3 where t > 2 and t < 4.
SELECT * from S4 where t > 3 and t < 6.

Hence, this super-query–set generated will minimize the volume of query/result transmission in the network.

4. PERFORMANCE EVALUATION

In this Section, the details about the simulator is presented, evaluation of the query processing system with bit-map and without bit-map scheme is performed, and the simulation results are discussed.

4.1 Simulator

A simulator has been used to evaluate the effectiveness of the bit-map approach. The simulator is developed using Java technology. The simulator works with few parameters in the network to be constant (like transmission range). The simulator is built to work for network with varying sizes. The sensor nodes are considered to form a fixed/static routing tree structure. The transmission range for each sensor is fixed as 10. Each of the sensors is initialized with power 100Joules. The sensors are employed for sensing temperature. The dataset used here is obtained from Intel Berkeley research lab data [10]. The query set is gathered from user experience. The simulator is limited to have input queries varying in their time intervals to reduce the complexity. Initially there is no cached data at the BS. As queries request data from the network, as and when the result is received, the BS stores a copy of the same, which becomes cached data.

4.2 Experimental results

A set of experiments were conducted to show the performance of the bit-map approach. The comparison is made based on the following metrics.

1. The Number of packets transmitted in the network for '*n*' queries.
2. The Lifetime of the network.
3. The Number of queries executed in the lifetime of the network.

The experimentation was done with networks of varying sizes. The presented reading for each performance metric is the average of the five simulation runs conducted on the simulator. The metrics computed are as follows: (i) number of packets transmitted, (ii) network lifetime and (iii) the number of queries executed in each experiment. The average values of the experiment results are depicted in the graphs presented in this section. First the network is simulated without bit-map approach and then using bit-map approach. In each experiment, the same set of 1000 queries is used to compute the number of packets transmitted in each of the networks. From Fig 5, it is observed that for executing 1000 queries in the network, the bit-map approach has shown a good decrease in the number of packets transmitted. From the results it is evident that as the network size increases the number of packets transmitted in the network decreases.

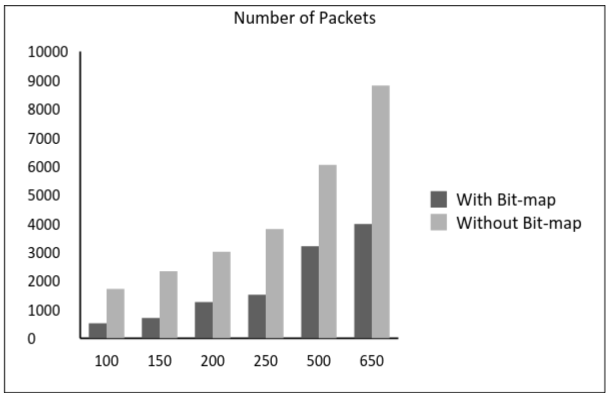

Fig 5: Graph showing the No. of packets transmitted in the network for executing 1000 queries.

As the network spends maximum energy for packet transmission, battery power is reduced after each transmission at every node. Hence, optimal battery power utilization is one of the major issues in WSN. The lifetime of the network for varying network size is presented as a graph in Fig 6. In the bit-map approach, the size of the super-query-set disseminated into the network is obviously smaller than the incoming queries at the BS. Hence, the network power is conserved, which increases the network lifetime. The graph in Fig 6 confirms this speculation.

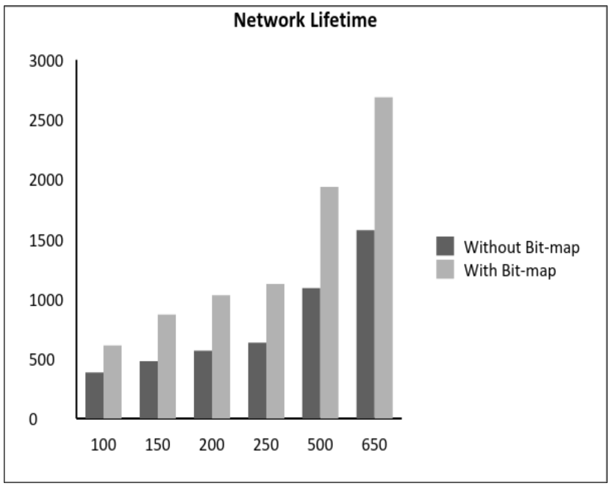

Fig 6: Graph showing the network lifetime.

The next experiment quantifies the number of queries executed during the network lifetime. The graph in Fig 7 shows the number of queries executed in the network of size varying from 100 to 650. There is a steep increase in the number of queries executed during the lifetime of the network w.r.t. the size of the network. Also, the graph shows higher number of queries executed in the bit-map approach.

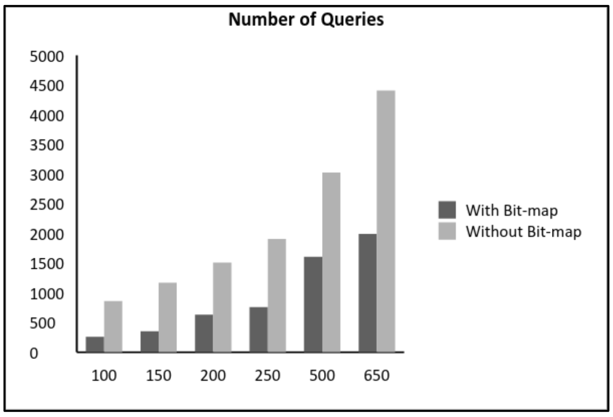

Fig 7: Graph showing number of queries executed during the lifetime of the network

From the above experimental results, it is clear that the bit-map approach results in optimal power utilization. In addition it is observed that effectiveness of the approach increases with the network size. Hence, it can be concluded that bit-map approach is successful in conserving the network power and increasing its lifetime.

5. CONCLUSION

The novel query processing technique presented in this paper exploits cached results at BS and query containment among the ad-hoc queries to be executed in the network. If the query results can be obtained in its entirety from the cached results then execute such query against the cached results at the BS and send the results to the user. Otherwise if the query results need to be extracted partially or completely from the network nodes, then the commonalities w.r.t., the data requirements of all such ad-hoc queries need to be identified to formulate super-queries, and transmit the same to the required nodes for execution. A bit-map approach is used for identifying the commonalities (containment) among the queries and to formulate super-queries. This helps in minimizing the number of queries executed and the volume of data transmitted in the network which eventually leads to optimal utilization of the power and longer network life. The focus of the query processing scheme in this paper was on devising effective approaches for processing ad-hoc queries. Nevertheless same techniques also can be applied on continuous queries as well. The experimental results prove the effectiveness of the proposed approach.

6. REFERENCES

[1] Akylidiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirici, E.: The survey on sensor networks. IEEE Communications Magazine 40(8), 114–120 (2002).

[2] Castelluccia, C., Mykletun, E., & Tsudik, G. (2005, July). Efficient aggregation of encrypted data in wireless sensor networks. In Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on (pp. 109-117). IEEE.

[3] Brayner, A., Lopes, A., Meira, D., Vasconcelos, R., & Menezes, R. (2008). An adaptive in-network aggregation operator for query processing in wireless sensor networks. Journal of Systems and Software, 81(3), 328-342.

[4] Yang, Chi, and Rachel Cardell-Oliver. An efficient approach using domain knowledge for evaluating aggregate queries in WSN, Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on. IEEE, 2009.

[5] Benzing, Andreas, Boris Koldehofe, Marco Volz, and Kurt Rothermel, Multilevel predictions for the aggregation of data in global sensor networks, In Distributed Simulation and Real Time Applications (DS-RT), 2010 IEEE/ACM 14th International Symposium on, pp. 169-178. IEEE, 2010.

[6] Behzadan, Afshin, and Alagan Anpalagan, Optimization of multiple overlapping queries for energy efficient sensor communication, In Communications (QBSC), 2010 25th Biennial Symposium on, pp. 181-186. IEEE, 2010.

[7] Chen, Tao, Nong Xiao, and Fang Liu, Multi-aggregate-query scheduling over data streams, In Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2010 International Conference on, pp. 27-33. IEEE, 2010.

[8] Müller, René, and Gustavo Alonso. "Shared Queries in Sensor Networks for Multi-User Support", ETH, Department of Computer Science, 2006.

[9] Huei-You Yang, Wen-Chih Peng and Chia-Hao Lo, Optimizing Multiple In-Network Aggregate Queries in Wireless Sensor Networks, Advances in Databases: Concepts, Systems and Appliations. Springer Berlin Heidelberg, 2007. 870-875.

[10] Intel Berkeley Research lab, http://db.csail.mit.edu/labdata/labdata.html.