XQuery based Query Processing Architecture in Wireless Sensor Networks

Manish Kumar
IIIT, Allahabad
India

Monalisa Panigrahi
IMS College, Ghaziabad
India

Shekhar Verma
IIIT, Allahabad
India

ABSTRACT

A wireless sensor network (WSN) deployed to sense an environment should be able to send information as per the requirements of the sink. For this, the sink needs a query language to formulate its needs and fire queries to the network. The queried nodes must be able to process the multiple queries in their limited memory and computational capacity in a time bound manner. In this paper, an XQuery based query processing architecture that satisfies the unique needs of WSN has been proposed and evaluated. Traditional SQL does not satisfy the time limitations, source specificity and node constraints required by a WSN. X-query architecture allows spatial aware queries that can be processed by resource constrained nodes. TOSSIM based simulation of X-query processing is energy efficient and the nodes are able to process multiple queries within a small time period.

General Terms

Query Processing, Sensor Networks

Keywords

WSN, Query Processing, X-query, SQL

1. INTRODUCTION

A wireless sensor network (WSN) is distributed system of wireless sensor nodes. These small sensors measure environmental and physical properties like temperature, pressure, humidity [1, 2] and organize themselves to deliver this data to a common data collector sink over the wireless medium. A sensor node is a low cost small hardware device suited for large scale deployment and autonomous operations. The cost, physical size, deployment regions, operational requirements and wireless communication medium restrict the memory size, processor capacity, operating platform, energy and reliability etc. of the nodes and the sensor network.

The sensor network can be considered as a network database [3]. Such a logical database is a distributed database which contains a large number of nodes that contain environmental data and respond to user queries. The data sensed by sensor nodes have location specific significance. If the location of the occurrence of an event observed by a sensor node is not known, the data about the event is meaningless. Moreover, the data reported by sensor nodes are largely raw. Thus, there is a need for a query system [4, 5] and distributed query processing [6, 7] that requires sophisticated processing of data generation and storage of distributed data. The wired distributed networks have high capacity storage, continuous power supply and high end processors required for high performance data manipulation. A WSN is at the lowest end of distributed networks with differentiation in terms of power, communication, storage etc. Hence, alternative ways have to be devised for querying, data gathering and other unique requirements.

A middleware [8, 9, 10, 11] is used as an interface between high-level abstraction and system-level programming concept in WSN. It can be used in such applications to hide the heterogeneity and distributed nature of the networks. The middleware [12, 13] is a software that can be used to fill the gap between the user applications and low level constructs. An important middleware used in query processing in WSN is TinyDB [15]. TinyDB provides a single unified declarative query (such as SQL) interface for the ease of communication between user and sensor networks. A user can fire Tiny-SQL queries (similar to SQL queries) to the base station. These queries are distributed to all the relevant nodes in the WSN from the base station that gather the result and send the response back to the user. Due to limited resources, Tiny-SQL does not support all type of SQL queries. The traditional TinySQL has several limitations. It does not give optimized result in multiple query scenarios. TinyDB uses a distributed in-network query processing architecture. The components of the TinyDB are built on top of TinyOS [14]. The user query should pass through these entire components [15].

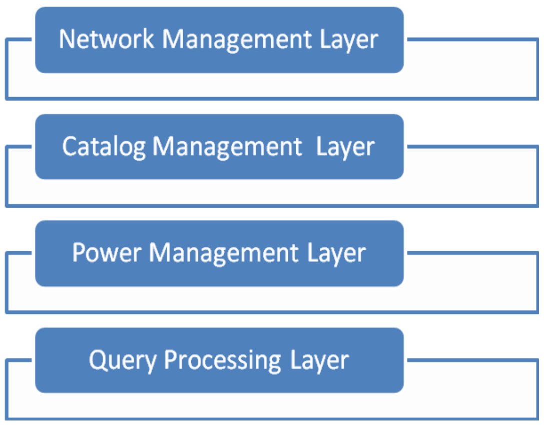

Fig. 1 Query Processing Layers

The different layers of the query processing as shown in Fig. 1:

i) *Network Management Layer*: The function of this layer is to improve query performance in multihop network. This layer also supports query dissemination [16] and query sharing. It helps in reducing the cost of the network. ii) *Catalog Management Layer*: This layer is used to store commands, events and attributes. It provides TinyDB a single unified interface to interact with TinyOS components. iii) *Power Management Layer*: The main function of this layer is scheduling. TinyDB uses slotted approach scheduling technique. iv) *Query Processing Layer*: After the query dissemination, the role of query processor starts. It consists of two phases: Preprocessing phase and Execution phase. The former is responsible for the testing of validation of the query; and the latter is responsible for query operation and the result gathering. The system is in idle state after execution phase until the next epoch starts.

In the present study, an alternative way of processing multiple queries is proposed using XML based XQuery interface. An XML based application is efficient for data interchange and allows ease in search due to its hierarchical representation. An XQuery engine is implemented for query processing in sensor network. The rest of the paper is organized as follows. The problem of query processing in sensor networks is defined in section 2. The proposed X-query solution is developed in section 3. This proposed solution is evaluated and the results are discussed in section 4. The conclusions are given in section 5 of the paper.

2. PROBLEM DESCRIPTION

The database of the sensor network as a distributed database is beset by limited by processing and memory limitations of the nodes, the need of energy efficient database operations, the nature of data sensed by the sensors and the spatial specificity of data in the sensor network. Another problem with traditional middleware like TinyDB is the dependence on one interface, i.e., declarative SQL interface for query processing. SQL has its own limitation which limits the performance of a sensor network. It always depends on relational databases and hence dependent on the system. It does not give optimized result in multiple query scenarios, and query interface also does not support heterogeneity in wireless sensor networks.

3. PROPOSED SOLUTION-XQuery

The traditional query processing in WSN uses SQL query interface. The user sends Tiny-SQL queries to the server. Due to limitation in SQL, the query processing system has three main limitations: i) SQL works well with relational model and hence SQL query returns only tabular data. The structure of a sensor network is hierarchical and need a query language that can traverse in a tree manner; ii) It is dependent on the underlying components; and iii) This system does not give optimized performance for multiple queries. The drawbacks present in the system motivated for alternative way, namely '*XQuery*' for execution of query which is coming continuously and also for the multiple query request with optimization. The advantage of XQuery over SQL is the first step towards the development of XQuery based query engine for query processing in sensor network. The limitation listed above is as addressed by XQuery are as follows.

- XQuery is very flexible in nature. It can query both tabular and hierarchical data, therefore, it is suitable for the sensor network.
- Extensibility – Xquery supports user defined tags depending on the application.
- XQuery is independent of underlying software.

3.1 XQuery based Architecture

A query interface is proposed to the user so that the user can interact with the WSN with ease. Fig. 2 illustrates the XQuery query processing architecture for sensor networks. The proposed approach mainly focuses on efficient execution of multiple queries in WSN. The user sends the request in the form of XQuery to the server and gets the response from the sensor network.

The XQuery engine is divided into three modules as client side, base station and sensor network. The functionalities of each module are as follows:

Client Side: This module contains all the clients or users, named as XQuery Client. The administrator can restrict the number of clients while running the server or base station. The client will send the query in XQuery, named as XRequest to the server, known as base station. The client will get the result back from the base station in XML format, named as XResponse.

Base Station: This module acts as an intermediate node between the client and a WSN. The base station is configured in a high processing system which will act as server for the whole system. The main implementation is done on the base station. A base station is responsible for the following tasks:

- Receive the XRequest from clients
- Convert the XRequest into corresponding temporary SQL query
- Send the temporary SQL query to the WSN
- Receive the response from WSN
- Convert the response in XResponse.
- Send the XResponse to corresponding clients.

The base station has four sub modules:

i) Request Listener: The function is to listen the entire client's request through sockets which is already connected to the base station.

ii) XQuery Parser: It takes the XRequest as an input from the Request Listener. The function of XQuery parser is to convert the XRequest into corresponding SQL if required. The parsing is done using Java programming and the generated temporary SQL query is injected into the sensor network.

iii) SQL Server Database: The SQL server is a part of base station. In sensor network, it is not possible to store all the data coming for multiple nodes within the network itself and therefore an intermediate SQL server is fixed which is placed between the base station and sensor network in order to provide buffering for multiple queries coming from multiple clients. When client sends XRequest, the SQL server matches request with its stored field, if a match occurs, then the result is sent to the SQL parser.

iv) SQL Parser: This component receives the temporary responses from the sensor network via SQL server carrying the responsibility of converting the responses into XQuery, known as XResponse and forwarding it to the corresponding client.

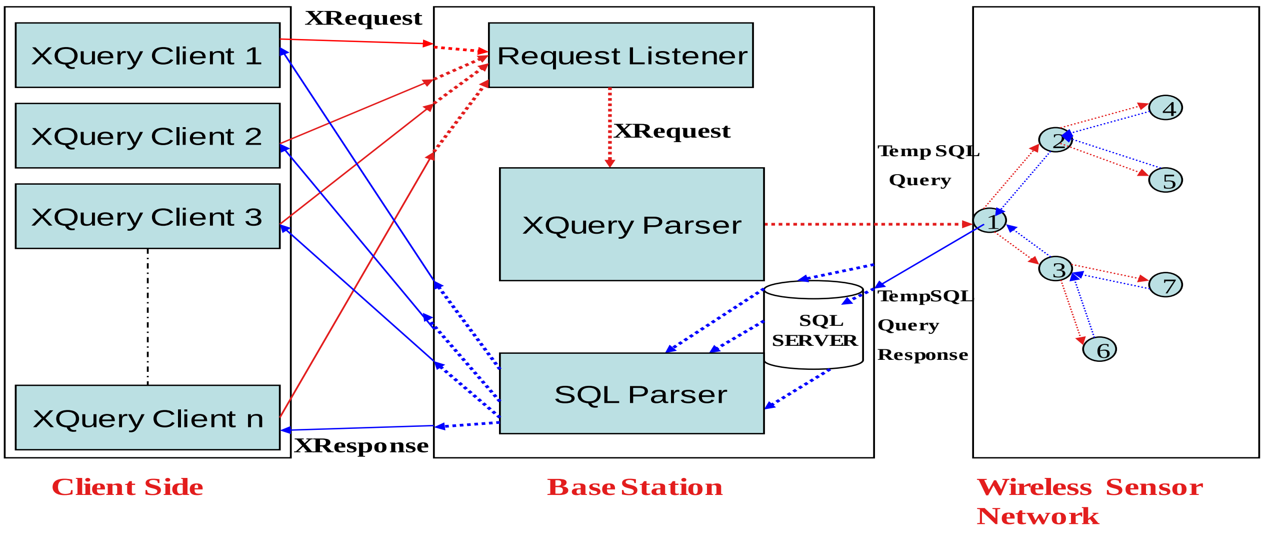

Fig.2 XQuery Query Processing Architecture

Sensor Networks: The whole WSN is organized as hierarchical network with multiple layers and each layer is having one or more parent nodes. One node elected as root node responsible for taking the temporary SQL query as input and passing the same to each and every child nodes. Each node creates a data packet containing node ID and sensed data. The node ID of root node is '0' and other nodes have node IDs 1, 2, 3, … etc.

3.2 Design and Development

The design of XQuery engine is divided into three phases:

Phase I: The first phase includes XQuery client-server program generation. The client-server communication (Fig. 3) is established using socket programming and multithreading. How many XQuery clients will be connected will be decided by the server or base station. Each XQuery client is considered as a separate thread in the program. More than one client can connect to the server simultaneously. The server listen the client request through sockets and once the connection is established, the XQuery client sends the XRequest to the server to get the data from the sensor network.

Phase II: It implements RequestListener in ServerThread class which listens all the XRequests from XQuery clients. The XRequest is passed to the XQuery parser. Then, the SQL parser converts back to the result from the sensor network into XResponse and sends back to the respective client. In the server-side implementation, all main threads are stated to initiate the server. Whenever a client connects to the server, the corresponding client thread is started. One special thread for Cygwin is started as well for TOSSIM simulator [90, 91]. The overall implementation is shown in Fig. 4.

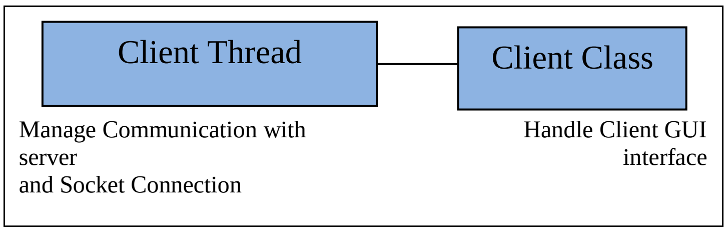
Client Thread
Client Class
Manage Communication with server and Socket Connection
Handle Client GUI interface

Fig. 3 Implementation of XQuery Client

Example of Parsing XQuery into SQL:

The HandleClient class of the server package handles the parsing. Suppose the Client sends the XQuery as

```
for $x in db("sensor")/mica2  return $x/nodeid,
$x/light  $SIM$
```

The above query is converted to the SQL as:

```
select node_id, light from sensor
```

where sensor is the relation name for sensed data. If client wishes to submit multiple queries simultaneously, then XQueries must be separated by $#$.

Phase III: This phase deals with the communication between the base station and the WSN. Basically the communication is done through TinyDB API. To simulate the above scenario, TOSSIM [17] simulator is used and need to integrate the java code to run in the Cygwin environment with the simulator and therefore added as separate threads and as a result automatically the TOSSIM simulator is started on the Cygwin console in the background.

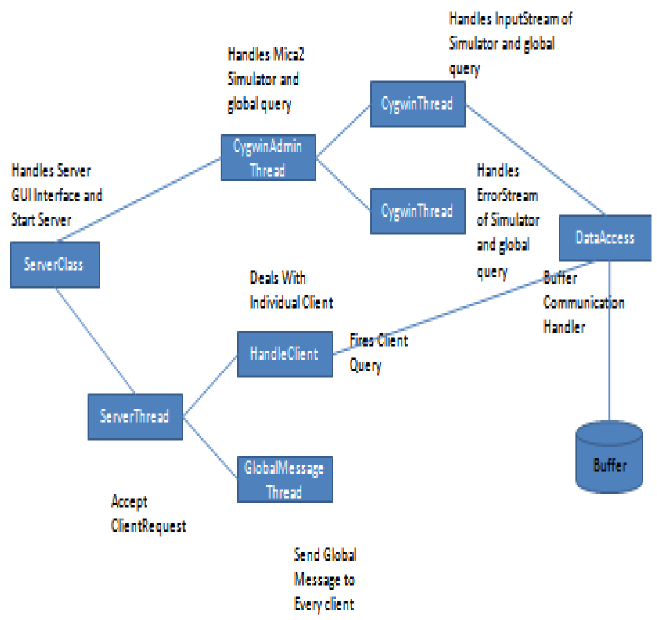
Handles Server GUI Interface and Start Server
ServerClass
Handles Mica2 Simulator and global query
CygwinAdmin Thread
Handles InputStream of Simulator and global query
CygwinThread
CygwinThread
Handles ErrorStream of Simulator and global query
DataAccess
Buffer Communication Handler
Deals With Individual Client
HandleClient
Fires Client Query
ServerThread
GlobalMessage Thread
Buffer
Accept ClientRequest
Send Global Message to Every client

Fig. 4 Implementation of XQuery Server

4. SIMULATION AND RESULTS

A client can submit multiple queries at base station. Moreover, multiple clients can send request for data of interest to the WSN. A socket program is executed to establish the client-server communication. A client sends multiple queries in form of XRequest to sever. A server sends back the results in form of ordered set of values as XResponse to clients. The simulation starts with motes configuration connecting to base station for data transfer. We have considered eight motes that are capable of sensing temperature and light and arranged in hierarchical WSN. Since the setup has been created on simulator, the temperature and light reading are recorded in uninteger 16 format. A user query is parsed and transformed at server and then directed to WSN. This process reduces energy consumption of sensor network significantly.

In Table 1, the output of client's queries is tabulated with node_id, light readings, temperature readings and timestamp of the nodes. It can be observed that not all nodes responding to all queries. However nodes are sending data based on client region of interest. We have injected multiple XQueries for different clients and gathered individual node responses. Table 1 is a snapshot example of the gathered data. The timestamp difference has been generated for the multiple clients for their multiple XQuery request. It is clearly sited in Table 2 that XQuery is capable of executing the multiple queries for multiple clients in a fast manner. The time interval between different XQuery clients shown in table is found suitable for querying in WSN. In Table 3, the time interval between different clients has been gathered to investigate the suitability of query processor. The results are generated and forwarded to clients in few milliseconds that show the suitability of XQuery for WSN. The time interval between client 1 and client 3 is noted (00.00.01.06). This may be due to the traffic density or delay in data sensing. Analyzing the performance, we observed that XQuery query processor reduces energy consumption and optimizes the multiple query scenarios in WSN.

5. CONCLUSION

A different approach for query processing in wireless sensor networks known as XQuery Engine was developed. The XML based XQuery Engine provides a unique query interface which suits the resource constrained wireless sensor network environment. In contrast to the other the traditional SQL query interface, XQuery Engine provides the client with the facility to fire multiple queries to the base-station. It also allows multiple clients to be connected to the server simultaneously. The results are obtained as an ordered set of values, known as XResponse. The performance analysis shows that the proposed XQuery Engine is lightweight and allows multiple query handling in an energy efficient manner.

6. REFERENCES

[1] Akyildiz I.F., Su W., Sankarasubramaniam Y., and Cayirci E., "A survey on sensor networks", IEEE Communications Magazine, vol. 40, no. 8, pp. 102-114, 2002

[2] Heidemann J., Fabio Silva F., Intanagonwiwat_C., Govindan R., Estrin D. and Ganesan D., "Building Efficient Wireless Sensor Networks with Low-Level Naming", In Proceedings of 18th ACM Symposium on Operating Systems Principles, 2001.

[3] Govindan R., Hellerstein M., Hong W., Madden S., Franklin M. and Shenker S., "The sensor network as a database", Technical Report-No. 02-771, 2002.

[4] Yao Y. and Gehrke J., "Query processing in sensor networks", In Proceedings of International Conference on Innovative Data Systems Research, 2003.

[5] Krishnakumar T., "Integrated Routing and Query Processing in Wireless Sensor Networks." International Journal of Applied engg. Research,Vol. 1, No. 1, 2010.

[6] Da Silva R.I., Del Duca Almeida, V., Poersch A.M., Nogueira J.M.S., "Spatial query processing in wireless sensor network for disaster management" Wireless Days (WD), 2nd IFIP, pp. 1-5, 2010.

[7] Yoon S. H. and Shahabi C., "Distributed Spatial Skyline Query Processing in Wireless Sensor Networks" In IPSN, 2009

[8] Fok C., Roman G. and Lu C., "Mobile agent middleware for sensor networks: An application case study", In Proceedings of 4th International Conference on Information Processing in Sensor Networks , pp. 382-387, 2005.

[9] Heinzelman B.W., Murphy A., Carvalho H., and Perillo M., "Middleware to support sensor network applications", IEEE Network, vol. 18, no. 1, pp. 6-14, 2004.

[10] Souto E., Guimaraes G., Vasconcelos G., Vieira M., Rosa N. and Ferraz C.," A message-oriented middleware for sensor networks", In Proceedings of 2nd ACM International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC), pp. 127-134, 2004.

[11] Ville St. L. and Dickman P., " Garnet: A Middleware architecture for distributing data streams originating in wireless sensor networks", In Proceedings of 23rd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 235-241, 2003.

[12] Hadim S. and Mohamed N., "Middleware challenges and approaches for wireless sensor networks", IEEE Distributed Systems Online, vol. 7, no. 3, pp. 1-23, 2006.

[13] Hadim S. and Mohamed N., "Middleware for wireless sensor networks: A survey", In Proceedings of 1st IEEE International Conference on Communication System Software and Middleware (Comsware), 2006.

[14] www.csl.stanford.edu/~pal/pubs/tinyos-programming

[15] Madden S., Hellerstein J. and Hong W., "TinyDB: In-network query processing in TinyOS", Version 0.4, 2003.

[16] Kulik J., Rabiner W. and Balakrishnana H., "Adaptive protocols for information dissemination in wireless sensor networks", In Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), 1999.

[17] Levis P., Lee N., Welsh M. and Culler D., "TOSSIM: Accurate and scalable simulation of entire tiny os application." In Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems, pp. 126-137, 2003.

Table 1 Query Results from Sensor Network

XQuery Client 1	Node Id **(uint16)**	Light **(uint16)**	Temp (uint16)	Timestamp (hr:min:sec: ms)	**XQuery Client 2**	Node Id **(uint16)**	Light **(uint16)**	Temp (uint16)	Timestamp (hr:min:sec: ms)
XQuery1	1	347	831	19:50:33.25	XQuery1	6	19	135	19:50:32.5
	8	578	919	19:50:33.11		7	21	950	19:50:32.423
	5	684	609	19:50:32.907		3	808	581	19:50:32.343
	6	19	135	19:50:32.5		5	1012	947	19:50:32.267
XQuery2	2		893	19:50:33.657	XQuery2	8		919	19:50:33.11
	3		442	19:50:33.58		5		609	19:50:32.907
	1		831	19:50:33.25		6		135	19:50:32.5
	8		919	19:50:33.11		7		950	19:50:32.423
XQuery3	4	705		19:50:33.953	XQuery3	1	347		19:50:33.25
	2	980		19:50:33.657		8	578		19:50:33.11
	3	221		19:50:33.58		5	684		19:50:32.907
	1	347		19:50:33.25		3	808		19:50:32.343
XQuery4	1		943	19:50:34.08	XQuery4	3		26	19:50:33.58
	4		698	19:50:33.953		1		993	19:50:33.25
	2		740	19:50:33.657		8		522	19:50:33.11
	3		31	19:50:33.58		5		548	19:50:32.907
	1		730	19:50:33.25		6		773	19:50:32.5
XQuery5	1	417		19:50:34.08	XQuery5	2	980		19:50:33.657
	4	705		19:50:33.953		3	221		19:50:33.58
	2	980		19:50:33.657		1	347		19:50:33.25
	3	221		19:50:33.58		8	578		19:50:33.11

Table 2 Time Interval for Client's Queries

XQuery Clients	**Time Interval b/w XQuery2 & XQuery1**	**Time Interval b/w XQuery3 & XQuery2**	**Time Interval b/w XQuery4 & XQuery3**	**Time Interval b/w XQuery5 & XQuery4**
XQueryClient1	00:00:00.407	00:00:00.296	00:00:00.127	00:00:00.00
XQueryClient2	00:00:00.61	00:00:00.14	00:00:00.33	00:00:00.077
XQueryClient3	00:00:00.233	00:00:00.080	00:00:00.564	00:00:00.203

Table 3 Time Interval between Client's Query Results

XQuery Clients	**Time Interval between Query Results (hr:min:sec:ms)**
XQueryClient1 & XQueryClient2	00:00:00.75
XQueryClient2 & XQueryClient3	00:00:00.31
XQueryClient1& XQueryClien3	00:00:01.06