Sensors **2011**, *11*, 11833-11855; doi:10.3390/s111211833

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

A Query Result Merging Scheme for Providing Energy Efficiency in Underwater Sensor Networks

Yunsung Kim [1] and Soo-Hyun Park [2,*]

[1] Korean Minjok Leadership Academy, 1300, Sosa-ri, Anheung-myeon, Hoengseong-gun, Gangwon-do, 225-823, Korea; E-Mail: jkkim008@hanmail.net

[2] The Graduate School of Business IT, Kookmin University, 861-1, Jeongneung-dong Sungbuk-gu, Seoul, 136-702, Korea

* Author to whom correspondence should be addressed; E-Mail: shpark21@kookmin.ac.kr; Tel.: +82-2-910-4559; Fax: +82-2-910-4519.

Received: 26 November 2011; in revised form: 7 December 2011 / Accepted: 15 December 2011 / Published: 20 December 2011

Abstract: Underwater sensor networks are emerging as a promising distributed data management system for various applications in underwater environments, despite their limited accessibility and restricted energy capacity. With the aid of recent developments in ubiquitous data computing, an increasing number of users are expected to overcome low accessibility by applying queries to underwater sensor networks. However, when multiple users send queries to an underwater sensor network in a disorganized manner, it may incur lethal energy waste and problematic network traffic. The current query management mechanisms cannot effectively deal with this matter due to their limited applicability and unrealistic assumptions. In this paper, a novel query management scheme involving query result merging is proposed for underwater sensor networks. The mechanism is based on a relational database model and is adjusted to the practical restrictions affecting underwater communication environments. Network simulations will prove that the scheme becomes more efficient with a greater number of queries and a smaller period range.

Keywords: underwater sensor network; query management; query result merging; relational sensor network database; conditional query; periodic query

1. Introduction

A wireless sensor network (WSN) is a wireless network of spatially distributed autonomous devices (sensor nodes) using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. [1-3] Each sensor node in a sensor network has functions to sense physical or environmental conditions, to process sensed data, and to send or receive the sampled data or processed results. Sometimes, a sensor node might be equipped with actuators to control itself or the environment.

Recently, applications of wireless sensor networks to underwater environments arose as a promising division of research among scientists due to the systems' various application possibilities [4-6]. Underwater sensor networks are designed for applications like pollution monitoring, disaster prevention, strategic surveillance, oceanographic data collection and offshore exploration. Unmanned vehicles with several underwater sensors can also be utilized for underwater resource investigation and collaborative underwater data gathering monitoring tasks [4]. Acoustic arrays and sensors can also be deployed for underwater ecosystem monitoring, in which transceivers can detect and track the number, health and location of aquatic creatures [7,8].

One important challenge with underwater sensor networks is enabling underwater communications. Wireless sensor network communication on land is conducted mostly through radio frequency (RF) wave propagation, taking advantage of its speed, long range and reasonable energy expenditure [9]. For underwater communications, however, RF waves are inappropriate due to several reasons such as high dissemination rate, immense propagation power [10,11], and high exposure to communication interference due to low and restricted transmission bandwidth [12,13]. Therefore, efforts have been undertaken to apply acoustic waves as a means for underwater sensor network communication [12,14].

Underwater acoustic wireless sensor networks, however, still display severe flaws that hinder their utilization for comprehensive and exhaustive communication tasks. First, acoustic waves are much slower than RF waves and can travel at only 1.5 km/s, so real-time data transmission and time synchronization are very difficult [4]. Also, pulsing an acoustic signal underwater requires more transmission power than for terrestrial RF signals [4]. Lastly, due to the high bit error rate, a greater amount of data transmission incurs a proportionally large amount of data failures [14].

Therefore, we face an urgent need to reduce the amount and frequency of data transmissions, especially in underwater circumstances. Some studies on data storage sought to reduce redundant data conveyance by allowing users to selectively acquire data by querying the network [15]. One of the two major approaches to data storage in sensor networks is the "warehouse" approach, in which all sensor data is collected into a large, central database from which users can acquire their data of interest [15]. Another approach is the distributed approach, in which each distributed devices such as sensor nodes can act like a database and users query each node for data [16]. The distributed approach effectively reduced the redundant network traffic caused by the warehousing system [17]. With recent enhancements in ubiquitous technologies, more smart devices (smart phones, smart dust, smart tabs *etc.*) are expected to be utilized for real-time and user-friendly seismic monitoring, equipment monitoring and remote controlling, leak detection, and support for robot swarms through this approach [18].

However, such a query-based sensor network management strategy can only be effective when a few users are online. When many users query the whole network through ubiquitous data management systems, responses to some queries may contain the responses for other queries, in which case redundant query responses may engender problematic waste of energy and traffic and cause high possibility of bit errors [19]. Energy is extremely wasted, especially when the node has to transmit the headers attached to every additional message it sends. To reduce excessive data transmissions, query aggregation and other researches on enhancing query processing were conducted [17,19], but there still is a great lack of studies on managing condition-based queries (request for data that satisfies query conditions).

In this paper, we suggest a multiple query result merging scheme as a practical solution for the above problems in underwater sensor network data transmissions. The proposed framework is a technique for providing data on-demand from an underwater sensor network to multiple-users simultaneously in an energy-efficient manner. Our paper's main focuses are:

(1) A relational database model view suitable for underwater sensor network applications

A relational underwater sensor network database model view is proposed. In conventional sensor network database models, time synchronization is guaranteed. Because time synchronization is very difficult in underwater sensor networks, however, we employ the "current relation", a single tuple relation of current sensor data, in our model. The queries request periodic data transmissions from these current relations. This model view allows us to access the sensor data in a sensor network using relational database concepts.

(2) The Query Managing Theorem and Period & Duration Management schemes

We give a theoretical account of queries and the basic principles of our query result merging mechanism. On these grounds, we provide an efficient query result merging mechanism. We also provide a new methodology for managing a number of continuous queries with different periods and durations.

(3) The comprehensive query management framework and a new payload format

A complete query management framework is thoroughly proposed. The management framework is categorized into four phases: Query Transmission Phase (QTP), Query Processing Phase (QPP), Query Response Phase (QRP), and Response Distribution Phase (RDP). A new payload format suitable for our proposed query management framework is devised and introduced.

The rest of this paper is organized as follows: Section 2 reviews the current research *status quo*, Section 3 explains the basic network and query models in discussion, Section 4 provides a mathematical approach to query result merging and its theorems, Section 5 comprehensively explains the four steps of query management framework, and Section 6 sets up a simulation model for performance evaluation, simulates the proposed framework and discusses the results. Section 7 reorganizes our contributions, suggests future work and concludes the whole paper.

2. Related Work

Several projects have introduced diverse mechanisms for distributed data querying, some of which include the Cougar system and the TinyDB. The Cougar system is a query processing mechanism that consists of three layers: front end, cluster leader, and the cluster node [16]. Each cluster node keeps its sensor data in the form of a time-series [20]. When the user sends a query to the network, the queries are transmitted to the front end, a cluster leader, and eventually delivered to the cluster node, which transmits the queried data in the reverse manner [16]. The TinyDB is a query processing system for extracting information from a network of TinyOS sensor nodes. When the user requests certain data with an SQL-like query language, the nodes take readings from the current node and merge it with sub-aggregates from the subtree, returning the node's aggregate data up the tree [21].

Both of these systems, however, lack query optimization schemes, and thus, are very vulnerable to heavy network traffic caused when many users submit multiple queries simultaneously. To alleviate network traffic and reduce energy waste, many researchers have studied data aggregation schemes as in the directed diffusion methods studied by [22] or the data centric routing schemes proposed by [23]. Several studies on effective data routing protocols and localization architectures were conducted by [19,22,24-27] as well. However, these researches were focused on efficient data handling without much focus on query or query result managements.

Some of the most closely related works on query management are [19,25]. Document [19] suggests a query aggregation mechanism for location-based queries. In [19], a multi-layer query aggregation method was introduced, in which the query manager aggregates queries and sends the merged query to each of the access points, where the queries are distributed among the target nodes. This project, however, only takes snap shot queries into account and disregards continuous queries (requests for data transmissions over a certain period of time).

In [28], a concept of condition-based query merging similar to our approach is introduced. The project, however, is immature and lacks theoretical grounds. Furthermore, in [19,28] no management methodologies for continuous queries are provided and their naïve approach to merging queries bears significant flaws and inefficiencies that are addressed and resolved in our paper. The project also lacks a generalized query processing scheme.

3. Underwater Acoustic Sensor Network Models and Query Models

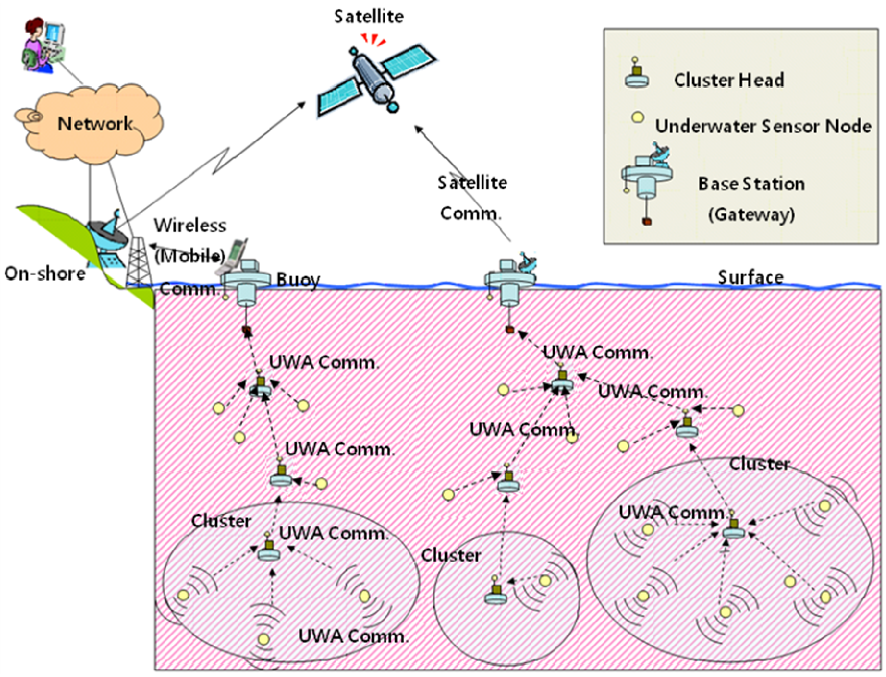
In this section, we explain the informational background of underwater sensor networks. Then we explain our network database view and query model.

3.1. Underwater Acoustic Sensor Network Backgrounds

As explained in the introduction, our main focus is to provide practical and efficient data management framework for underwater sensor networks. An underwater sensor network usually consists of sensor nodes, clusters and base stations. Each terminal node transmits the sensor data toward the cluster header. The cluster header collects sensor data from a number of terminal nodes and tosses the packet up the tree. Using certain routing mechanisms, these intermediate nodes transfer the packet to each other and eventually deliver the packet to a base station placed at the surface, where the acoustic data signals are transformed into corresponding RF signals and transmitted to the user applications [23,24]. Figure 1 shows the underwater acoustic sensor network model.

One major disparity between the common RF based sensor networks and underwater acoustic sensor networks is time synchronization. As explained in the introduction, low signal propagation speed and limited human accessibility to underwater sensor nodes makes time synchronization very difficult. In our network model, therefore, sensor nodes are assumed to execute queries individually without any agreement on time, and the following network database and query models are modified on such basis.

Figure 1. Cluster-based underwater acoustic sensor network model.



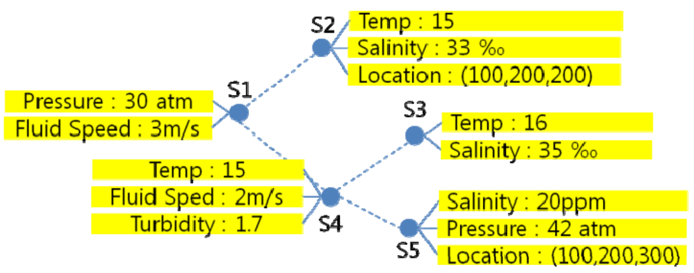
3.2. Relational Underwater Sensor Network Database Model

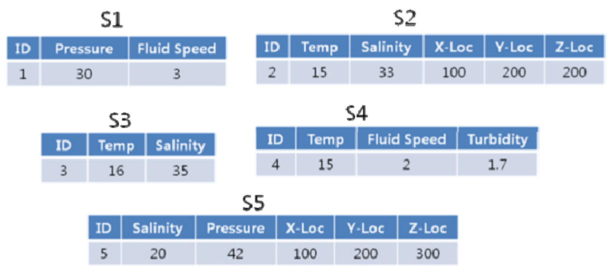
For better sensor network data management, many attempts have sought to apply database models to sensor networks. Relational databases serve as useful tools for systematized sensor network data management because of their simple data structure, suitability for queries, and simple manipulation languages [20]. In our approach we follow a similar concept.

Any sensor network can be mapped to a relational database model. In our data model, each sensor node represents a relation, where each sensor and node ID is the relational attribute and the sensor value range is the attribute domain. Each node memory contains a number of ordered lists of sensor data at a certain instant. Each list is mapped to a tuple in a sensor node relation, and the collection of these relations is the sensor network database.

In some of the relational sensor network database models, a sensor node keeps a partial history of recent sensor data. In these models old tuples are constantly updated and replaced by new ones, and queries extract data only from the remaining tuples. However, because underwater signal propagation is significantly vulnerable to water environments, there is an ample possibility that the requested data has already been deleted from the node relation before the query reaches the node. Also, difficulties in time synchronization make sensor node data timeline ineffective for underwater acoustic sensor network applications. Therefore, our network database model employs a single-tuple relation of current sensor data called "current relation". The current relation is updated whenever queries are assigned. For example, the sensor network in Figure 2(a) can be mapped to a corresponding relational database in Figure 2(b).

Figure 2. Example of a relational underwater sensor network database model. The sensor network in **(a)** can be mapped to a relational database model in **(b)** where each of the sensors is mapped to one of the current relations.




S1

ID	Pressure	Fluid Speed
1	30	3

S2

ID	Temp	Salinity	X-Loc	Y-Loc	Z-Loc
2	15	33	100	200	200

S3

ID	Temp	Salinity
3	16	35

S4

ID	Temp	Fluid Speed	Turbidity
4	15	2	1.7

S5

ID	Salinity	Pressure	X-Loc	Y-Loc	Z-Loc
5	20	42	100	200	300

(a) **(b)**

3.3. Query Models

Users frequently request on-demand data transmissions using queries. Most of the queries are condition-based selection queries (requests for data transmissions under certain conditions), and projection queries (requests for transmissions of certain attribute values of sensor data). Users can also demand periodic data transmissions and stop the responses after a certain period. Based on these observations, all queries can be represented by the following model:

$$Q = o|^{T}_{(S,\, D)}$$

where:

O = query operation, a one-time data request (a finite combination of selection and projection queries: $o \in Span <\sigma, \Pi>$

T = data reply period in terms of unit time

S = query reply start time represented by the number of time units after the nodes query set start time (for a detailed explanation of query set start time, refer to Section 5.3.1.)

D = query deadline also represented by the number of time units after the query set start time

The queries' temporal information (period, start time and deadline) is arranged in terms of a unified atomic time unit. The specific length of the time unit is determined prior to all query operations. The following are two examples of our query model.

Example (1)

Q_1: Transmit the pressure value for sensor S5 every 2 s for 30 s (unit time: 1 s).

$$Q_1 = \Pi_{pressure}\left(\sigma_{ID=S_5}\left(S\right)\right)\Big|^{2}_{(0,30)}$$

Variable S refers to a dummy relation variable for a sensor node relation.

Example (2)

Q_2: Transmit the pressure and temperature values every 3 s for 15 s if the temperature is above 20 degrees and the depth is greater than 50 meters (unit time: 1 s).

$$Q_2 = \Pi_{pressure,temp}\left(\sigma_{temp>20 \wedge z-loc>50}(S)\right)\Big|_{(0,15)}^{3}$$

When we refer to a "query", by definition it refers to the periodic data transmission request over a certain period of time. We hereby note, however, that in the following sections, the notation "query" may also frequently refer to a one-time data request in the continuous transmission task at some moment (or simply, a query operation), as long as no specific temporal information is mentioned and there is no ground for confusion. For example, "responses to queries q_1 and q_4" would refer to the relations resulting after the two query operations are executed at a given time frame.

4. Theoretical Approach to Query Result Merging

In this section, we explain and prove two major theorems useful for query result merging along with important definitions.

4.1. The Query Reduction Principle

In this section, we explore the properties of queries and define query result merging. Following is a definition of a well-defined query.

Definition 1. Well-Defined Query

Let us denote that condition C "refers" to a set of attributes A when all the data fields involved in condition C is contained in set A.

Query:

$$q = \Pi_{A_n}\left(\sigma_{C_n}\left(\cdots\left(\Pi_{A_1}\left(\sigma_{C_1}(S)\right)\right)\right)\right)$$

is well-defined if condition C_i refers to a subset of attribute set A_{i-1} for every positive integer $1 < i \leq n$.

This definition is reasonable since selection $\sigma_{C_j}(\ldots)$ would be examining non-existent attributes if C_i does not refer to the attributes in A_{i-1} because projection $\Pi_{A_{i-1}}(\ldots)$ would leave out the attributes referred in C_i. Otherwise, all queries can be answered. The following theorem shows how all queries can be reduced to a simple form. Without loss of precision, we ignore the query temporal information for now.

Theorem 1. Query Reduction Principle

Every well-defined query:

$$q = \Pi_{A_n}\left(\sigma_{C_n}\left(\cdots\left(\Pi_{A_1}\left(\sigma_{C_1}(S)\right)\right)\right)\right)$$

can be reduced to the form:

$$q = \Pi_A(\sigma_C(S))$$

where $A = A_n$ and $C = \bigwedge_{i=1}^{n} C_i$

Proof

We prove this theorem through mathematical induction.

We first examine when *n = 2*. If *C* refers to *A*:

$$\sigma_C\left(\Pi_A(S)\right)=\Pi_A\left(\sigma_C(S)\right)$$

since selecting pairs and restricting them to set *A* is equivalent to doing the reverse as long as *C* refers to *A*. Therefore:

$$q=\Pi_{A_2}\left(\sigma_{C_2}\left(\Pi_{A_1}\left(\sigma_{C_1}(S)\right)\right)\right)=\Pi_{A_2}\left(\Pi_{A_1}\left(\sigma_{C_2}\left(\sigma_{C_1}(S)\right)\right)\right)$$

Note that:

$$\Pi_{A_2}\left(\Pi_{A_1}(X)\right)=\Pi_{A_2}(X)$$

because $A_2 \subset A_1$ and that:

$$\sigma_{C_1}\left(\sigma_{C_2}(X)\right)=\sigma_{C_1\wedge C_2}(X)$$

because the resulting tuples satisfy both C_1 and C_2. Therefore, $q=\Pi_A\big(\sigma_C(S)\big)$ for *n = 2*, where $A = A_2$ and $C=C_1\wedge C_2$.

Assume that the principle is true for *n = k*. Then for *n = k + 1*:

$$q=\Pi_{A_{k+1}}\left(\sigma_{C_{k+1}}\left(\cdots\left(\Pi_{A_1}\left(\sigma_{C_1}(S)\right)\right)\right)\right)$$

Since the nested operation: $\Pi_{A_k}\left(\sigma_{C_k}\left(\cdots\left(\Pi_{A_1}\left(\sigma_{C_1}(S)\right)\right)\right)\right)$ equals $q=\Pi_{A'}\big(\sigma_{C'}(S)\big)$, where $A' = A_k$ and $C'=\overset{k}{\underset{i=1}{\wedge}}C_i$, *q* can also be reduced to the form:

$$q=\Pi_{A_{k+1}}\left(\sigma_{C_{k+1}}\left(\Pi_{A'}\left(\sigma_{C'}(S)\right)\right)\right)$$

as examined in the case when n = 2. Hence:

$$q=\Pi_{A_{k+1}}\left(\Pi_{A'}\left(\sigma_{C_{k+1}}(\sigma_{C'}(s))\right)\right)=\Pi_A(\sigma_C(S))$$

where $A = A_{k+1}$ and $C=\overset{k+1}{\underset{i=1}{\wedge}}C_i$.

Query Reduction Principle is meaningful since it implies that all queries can eventually be reduced to its simplest forms that are much easier to deal with. On this foundation, we introduce our new query result merging methodology.

4.2. Query Result Merging Methodology

Here, we define query result merging and discuss its significance.

Definition 2. Query Result Merging

Let $\Sigma = \{q_i\}$ be a set of queries, where:

$$q_i = \Pi_{A_i}(\sigma_{C_i}(S))$$

Also, given a current relation S, let Σ_S be a set of queries in Σ whose conditions are satisfied by the tuple in S. Then the relation defined by:

$$M_\Sigma(S) = \Pi_{\cup_{q_i \in \Sigma_S} A_i}(S) \text{ (}A_i\text{: the set of projection attributes of query } q_i\text{).}$$

is called the merged query result for the query set Σ and the process of producing the merged query result relation is called "query result merging".

Logically, it is reasonable to call this "query result merging" because the process extracts only the data fields requested by the queries whose conditions are satisfied by the current relation S. Therefore, the resulting relation fully contains the requested data in the most compact manner.

Unlike in [28], where query conditions and requested data attribute sets were merged, we do not attempt to merge queries. Merging queries into a single query is dangerous because when query conditions and projection attributes are merged, the merged query would indistinctively request all data attributes, even for the queries whose conditions are not satisfied. Therefore, instead of incurring unnecessary data transmissions by query merging, we apply a more robust concept of query result merging. The following section explains the complete query management framework for underwater sensor networks in detail.

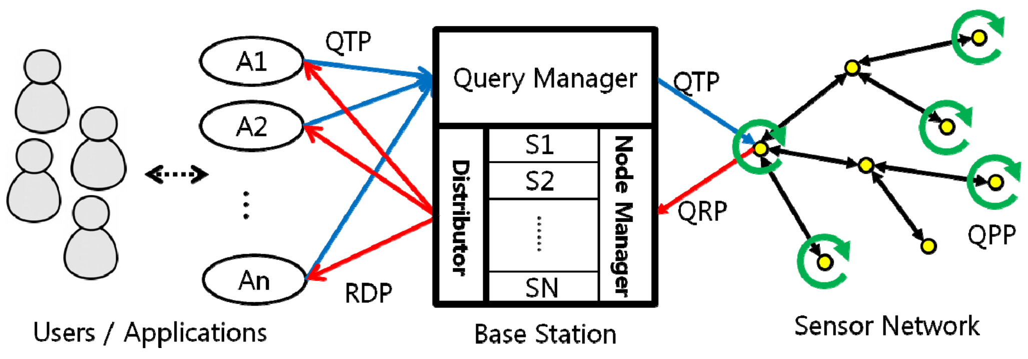
5. The Query Management Scheme

In this section, we briefly explain our framework settings and illustrate our proposed query management framework in detail.

5.1. System Architecture

Figure 3 briefly illustrates our system architecture, which consists of four major components: the users, the user applications, the Base Station and the sensor network. Below is a brief description of each component.

Figure 3. System architecture for the query management scheme.



(1) The Users/Applications

Each user uses an application to access underwater sensor networks. The users can send queries and receive query results through the base station with the user applications.

(2) The Base Station (BS)

In our system architecture, the base station plays a crucial role in organizing queries into sets, receiving sensor data from each node, reorganizing the query results, and transmitting the data to each of the applications. Since the BS assumes an extensive role, it is divided into three units, each of which is enumerated below. The BS has a comprehensive network metadata called a "network dictionary" that contains the information of sensors attached to each node and the location of node. With the network dictionary, the Base Station selects the "target nodes" for each query—the nodes that collect the data fields requested by each query. Figure 4 is an example of a network dictionary. The BS is provided with infinite processing/transmitting power.

(a) The Query Manager (QM)

The QM receives queries from the users up to as much as a node can process at a time, arranges them into a query set, and routes the set to targeted sensor nodes. The network dictionary is used to select the target nodes.

(b) The Node Manager (NM)

The NM receives the query result tuples from the sensor nodes. Using the time stamps on the result tuples, the NM generates a timeline of queries to determine which queries are responded at which moment. It then generates the final response tuple that has a number label to clarify its time sequence, and passes the differently labeled tuples to the Distributor.

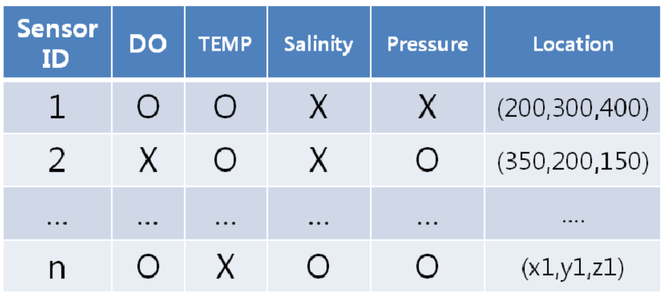
(c) The Distributor

The distributor receives the final tuples from the NM and sends the arranged query results to the user applications.

(3) Sensor Nodes

Each sensor node receives queries and merges the query results. It sends back result tuple to the BS.

Figure 4. Example of a network dictionary possessed by a BS.



Sensor ID	DO	TEMP	Salinity	Pressure	Location
1	O	O	X	X	(200,300,400)
2	X	O	X	O	(350,200,150)
…	…	…	…	…	….
n	O	X	O	O	(x1,y1,z1)

Our query management scheme is divided into four phases: the Query Transmission Phase (QTP), the Query Processing Phase (QPP), the Query Response Phase (QRP), and the Response Distribution Phase (RDP). The following subsections provide detailed explanations for each phase.

5.2. The Query Transmission Phase (QTP)

When users send queries to the QM, the QM sequentially assigns an ID to each query and goes through the network dictionary to determine which nodes to forward the queries to. The QM generates a collection of queries targeted to each sensor node (called a "query set") and routes these query sets to each of the target nodes. Queries are collected as much as each node can transmit at a time. If there aren't as many queries, the QM can periodically formulate a query set and transmit it to the network. In Figure 5, S_i, D_i, T_i (i = 1, 2,…, N) denote the query's start time, deadline and response period.

Figure 5. Example of a query set sent from the BS to a node.

Query ID	Query	S	D	T
1	O1	S1	D1	T1
…	…	…	…	…
N	ON	SN	DN	TN
SET Σ_1				

5.3. The Query Processing Phase (QPP)

The QPP is the core of our management scheme. In this phase, each sensor node receives its query set and merges the query results to generate the response message.

5.3.1. Query Set Start Time

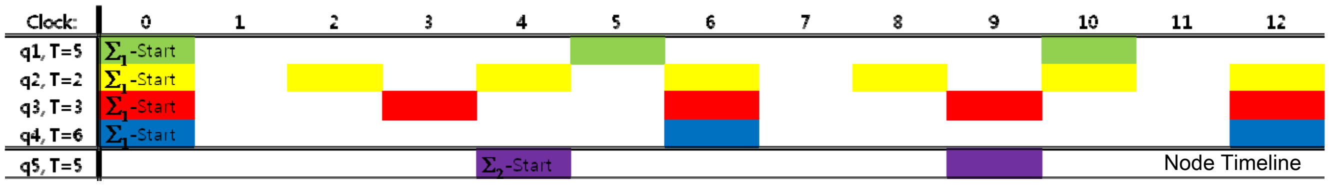
When the node receives a query set Σ_i, it automatically assigns a query set start time I_i, based on the time indicated by each node clock. The query set start time is the zero time point for all the temporal information within the set. The set start time for the first query set received by the node is 0. For example, let's say the QM sent two query sets, first Σ_1 then Σ_2, to a node as the following:

$$\Sigma_1 = \{q_1, q_2, q_3, q_4\} \text{ and } \Sigma_2 = \{q_5\}$$

$$q_1 = o_1 |_{(0,10)}^{5},\ q_2 = o_2 |_{(0,12)}^{2},\ q_3 = o_3 |_{(0,12)}^{3},\ q_4 = o_4 |_{(0,12)}^{6},\ q_5 = o_5 |_{(0,9)}^{5}$$

Figure 6 is a timeline of the queries sent to the node. The numbers on the first row indicates node clock and the colored boxes indicate query execution. For instance, when the node clock indicates 6, queries q_2, q_3 and q_4 must be conducted. In this figure, I_1 is 0 and I_2 is 4. The queries in Σ_2 are executed every 5 time units after the query set start time.

Figure 6. Timeline of queries sent to a node. The numbers indicate the time of the readings from the node clock. Each colored box indicates query execution.



5.3.2. Period and Duration Management Theorem

Given the list of queries, the node scans through the queries and their temporal information to select which queries to respond to at each moment. The following theorem provides a method of selecting queries for generating their responses at a specific time point.

Theorem 2. Period and Duration Management Theorem

Let Σ_t denote the set of all queries arrived at a node up to time t. Then, the set of queries waiting to be responded at t is defined by:

$$\Lambda_t = \left\{ q_i \in \Sigma_t \middle| t \equiv (I_i + S_i) \text{ mod } T_i,\ t \le (I_i + D_i) \right\}$$

where S_i, D_i, T_i and I_i are the start time, deadline, response period, and query set start time for query q_i.

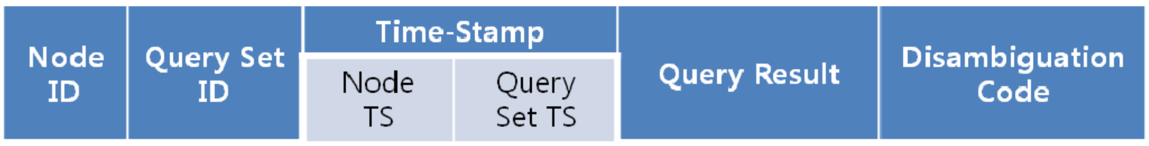
Proof

Because the query set start time is the zero point for the queries in the set, the first execution time for query q_i is $t = I_i + S_i$. Query q_i is replied every T_i since the first execution up to the deadline, which means queries are executed at $t \equiv (I_i + S_i)$ mod T_i while $t \le I_i + D_i$. Therefore, the merged query result at time t is $M_{\Lambda t}(S)$.

5.3.2. The Message Payload

With the query result tuple, the node creates a response message. Figure 7 is the message payload format for the node's application level. The data fields in the payload are explained below.

Figure 7. Response message payload format at the application level.



(1) Node ID: The node ID is the ID of sensor node that sent the response message.
(2) Query Set ID: The query set ID is the set ID of the most recently arrived query set. This field is used to identify the query set start time for different query sets.

(3) Time-Stamp: This field indicates when the response was made. Using the time-stamp, the Node Manager can generate a query timeline to determine which queries are executed at each moment. The Query Set time-stamp is added to adjust to message transmission failures, as explained in the Section 5.3.1.

(a) Node-Time Stamp: The node time-stamp is the time indicated by the node clock. The zero time point for the node clock is the query set start time of the first query set.

(b) Query Set Time-Stamp: The query set time-stamp represents the number of unit time after the start time of the most recently arrived query set. For example, if the latest query set arrived at Node TS = 14 and the current time is Node TS = 16, then Query Set TS = 2.

(4) Disambiguation Code (DC): This is a binary code included in case there is an ambiguity caused by multiple queries requesting the same data attributes under different query conditions. Each of the ambiguous queries is sequentially represented by each bit of the code. If the query condition is satisfied, the digit is set to 1; else the digit is clear.

For example, assume that the node has to execute the following three queries:

$$q_2 = \Pi_{temp}(\sigma_{pressure>20}(S))$$

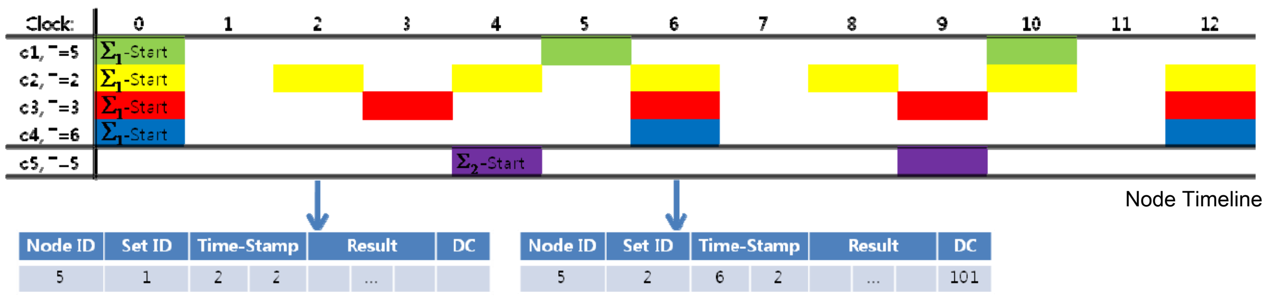
$$q_3 = \Pi_{temp,DO}(\sigma_{pressure<20}(S))$$

$$q_4 = \Pi_{DO}(\sigma_{temp<30}(S))$$

If the sensor data readings are: *pressue* = 30, *temp* = 20, the node sends both the temperature and the DO values in response to q_2 and q_4. However, with only the two field data, the Node Manager cannot recognize whether the pairs are responses to query q_3 (which also demands both the temperature and the DO values) or to queries q_2 and q_4. Therefore, a three-digit disambiguation code "101" is added to indicate which query conditions (q_1 and q_3 in our case) are satisfied.

Figure 8 is an example of two response messages at different time frames for a sensor node whose ID is 5. Notice that no disambiguation code is required for the first response message because there is only one query, which indicates no possibility of ambiguity.

Figure 8. Examples of response messages at two different points in time. The first response message has no disambiguation code because there is no possible ambiguity.



Node ID	Set ID	Time-Stamp		Result			DC
5	1	2	2		...		

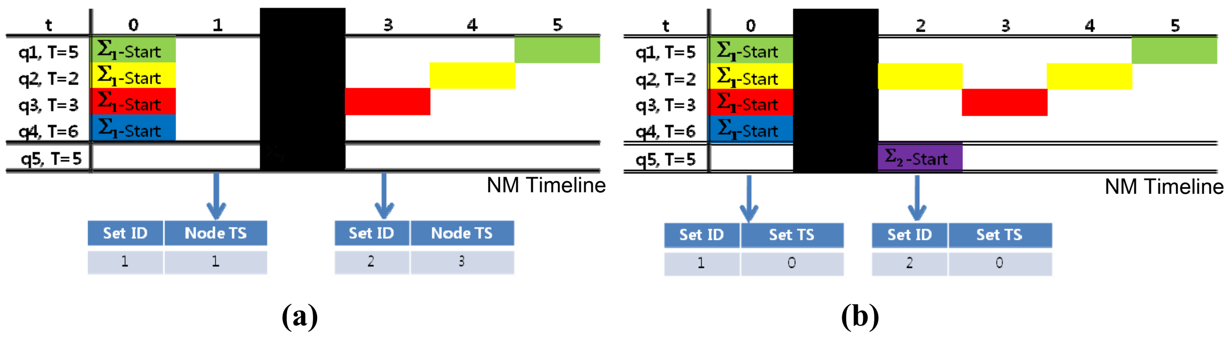
Node ID	Set ID	Time-Stamp		Result			DC
5	2	6	2		...		101

5.3.3. Robustness against Transmission Failures

Due to their hazardous communication environment, underwater sensor networks are highly exposed to communication failures. Especially when response messages are missing at the start of a new query set as in the following examples, a single time-stamp may cause critical mishaps to the NM when distinguishing query set start time.

In Figure 9(a,b), the original query set start time for Σ_2 is t = 2, but a response message is missing around the query set start time. In Figure 9(a), a response message is missing at t = 2. With only node time stamps as a time guide, the NM will mistake t = 3 as the set start time for Σ_2 because this is the first time Σ_2 appears in the NM. Similarly, in Figure 9(b) the message is missing at t = 1. The node time stamp is also be misleading since the NM would identify t = 1 as the start time for Σ_2 because Σ_2 first appears after the set time stamp changes from 0. Therefore, single time stamps are significantly vulnerable to communication failures and can influence the whole query management process.

Figure 9. Examples of how single time stamps can be misleading in case of communication failures. **(a)** The response message only contains node time stamp. **(b)** The response message only contains set time stamp.



With both the node and query set time stamps as time indicators, the NM can identify the query set start time by tracing the changes in the set ID and the node time stamp. The NM can also overcome intermittent message blackouts and keep track of exact time sequences.

5.4. The Query Response Phase (QRP)

In this phase, responses are transmitted from the nodes to the NM, which puts a number label to each of the tuples and passes the pairs to the Distributor. When the NM receives a response message, the NM can generate a query timeline (similar to the timelines in the above examples), using the time information elucidated by the response message, the start time for each query set, and each query's temporal information. It can also distinguish the sequence of the response tuples. The NM generates the final response tuple by extracting the requested data fields from the response message and putting the query ID, node ID, and a sequence number label to each of the extractions, and passes the final tuple to the Distributor. The final tuple is organized by the Distributor and sent to each user. Result tuples can be extracted by using the Data Extraction Theorem. The proof of the theorem is obvious, so it is not provided.

Theorem 3. Data Extraction Theorem

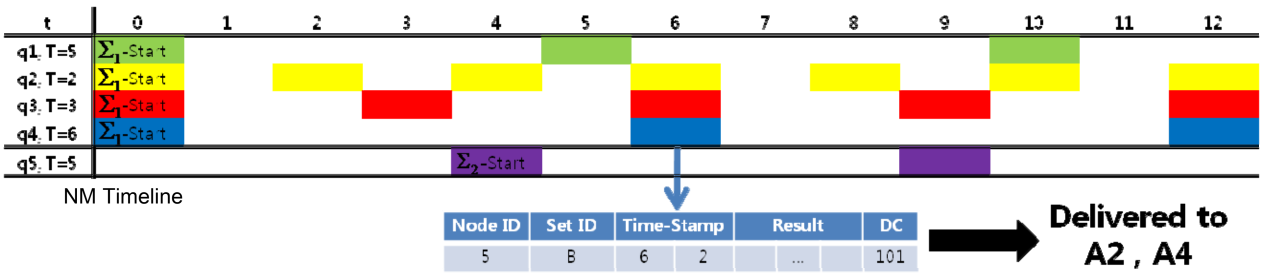
The NM can extract the data requested by query q_i from the result tuple by the following operation:

$$q_i(S) = \Pi_{A_i}(T)$$

where A_i is the data attribute requested by q_i and T is the result pair relation.

Figure 10 is an example of message processing at this phase. A_i is the user application that sent the query q_i:

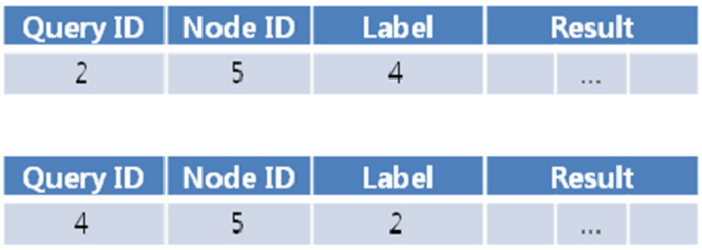
Figure 10. An example of message processing at the QRP. Analyzing the disambiguation code, the NM can determine that the queries should be sent to applications A2 and A4.



Node ID	Set ID	Time-Stamp		Result			DC
5	B	6	2		...		101

In this example, the NM generated a query timeline. When the NM receives a query response message from a node (sensor node 5 in this case), the node can identify which users to send the queries to. In our case, the disambiguation code is 101, which indicates that there are three ambiguous queries and the query results should be transmitted to the user applications for the first and last of the three (A2 and A4). In the final response messages generated for user application A2, the sequence number label would read "4" since it is the fourth response to query q2. Similarly, the sequence number label for user A4 would read "2". Final response messages are sent to each distributor would look like Figure 11.

Figure 11. Two examples of response messages delivered to the Distributor. **(a)** is delivered to A2 and **(b)** is delivered to A4.



Query ID	Node ID	Label	Result		
2	5	4		...	

Query ID	Node ID	Label	Result		
4	5	2		...	

5.5. The Query Transmission Phase (QTP)

This is the last phase of the query management scheme. In this phase, the Distributor receives queried data from the NM, and each query response is sent to the appropriate user applications. According to the users' demands, the Distributor can organize the queried data into time sequences, or it can simply deliver query responses whenever it receives a message from the NM.

6. Performance Evaluation

In this section, we describe our simulation model and evaluate the performance of the proposed management scheme.

6.1. Simulation Model

6.1.1. Network Model

Many studies on underwater sensor networks have proposed a cluster-based, vertical tree topology as an energy efficient network figure [29,30]. This is because the vertical tree topology reduces the number of hops required by the data for transmissions to the BS in underwater environments [31]. Thus, we also base our simulation on a cluster-based tree-topology sensor network. We assume that each node has the same number of child nodes, possibly except for the last parent node. Queries uniformly request data transmissions from all sensor nodes, and query responses are aggregated as query result messages are propagated up the tree. Every parent-child pair is assumed to be 1.5 kilometers apart (a second's distance for an acoustic wave). All queries and response messages are assumed to have average data sizes, and merged query result messages are to have larger sizes than normal query result messages (considering the additional data augmented to the payload). Additional data is added to the response payload as more queries are merged.

6.1.2. Energy Model

Our simulation used LinkQuest Inc.'s UWM1000 model for our underwater acoustic modem energy model. In this model, the payload data rate is 7 kbps, the transmission power 1 Watt, and the reception power 0.75 Watts [32]. Since the energy consumption for sleeping mode and data processing is much less compared to transmission/reception, we do not take these factors into account. We assumed that a node can consume up to 100,000 Joules of energy before it dies out.

6.1.3. Performance Metric

We analyze the average energy consumption (AEC) in terms of joules per node. The average energy consumption is the total network energy consumption divided by the number of sensor nodes in a network. We also observe the energy efficiency rate (EER) defined by the rate of the average energy consumption without merging to that with merging. For example, if the overall energy consumption was 6 kJ without the merging process and 2 kJ with the merging process, the EER is 3.

We also examine network lifetime and network lifetime ratio (NLR), defined by the rate of the network lifetime without the merging and with it. The network lifetime is the time duration until any one node in a network (which will be the sink node in our case since all data is transmitted through the sink node) dies out due to energy depletion.

6.1.4. The Simulation

The parameter for our simulation will be three variables: period range (T), number of queries (Q), and the number of nodes (N). Given a tuple of (T, N, Q), Q queries whose periods are uniformly distributed from 1 up to T are generated and executed for 10,000 time units. We will analyze the sensitivity of the AEC, EER, network lifetime and NLR to (1) the number of queries and (2) the period range.

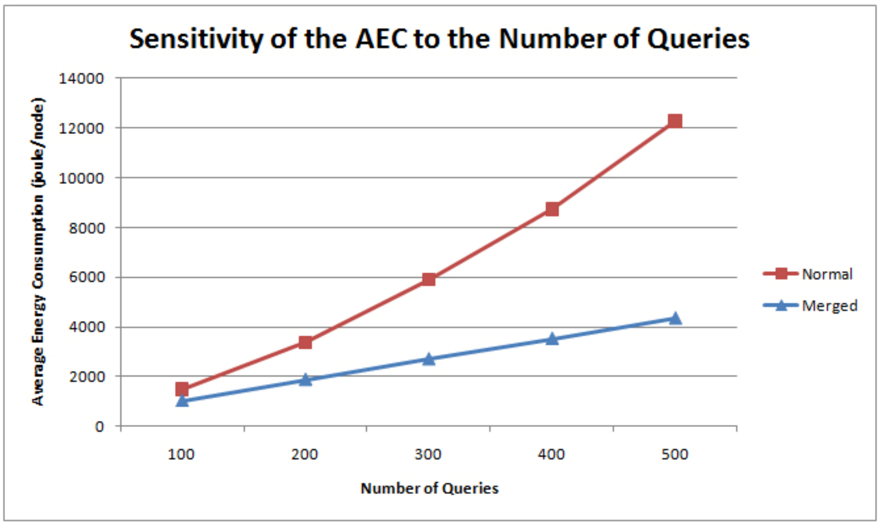
6.2. Simulation Results

In this section, we give an account of our performance results and discuss them.

6.2.1. Sensitivity to the Number of Queries

In the simulation, an increasing number of queries with period range 100 uniformly requested data transmissions from 50 nodes. Figure 12 is the graph of average energy consumption values for each node in a sensor network to the number of total queries.

Figure 12. Sensitivity of the AEC to the number of queries.



The graph indicates that with normal data transmission methodologies, the AEC increases slightly more rapidly as more queries are imposed on the whole network. This phenomenon can be attributed to the additional headers attached to the payload for each message transmission and the increased data aggregation at each parent node. In the proposed query management scheme, these redundant data transmissions are minimized and the AEC is kept low. The gradual increase in the AEC is due to the additional fields augmented to the payload at the Network and Data Link layers in response to more queries.

Figure 13 shows the relationship between the number queries and the energy efficiency rate (EER). Performance is improved as more queries were charged on the network. This effect is expectable since the proposed mechanism takes a greater advantage of query result merging when more users query the sensor network.

Figure 13. Sensitivity of the EER to the number of queries.

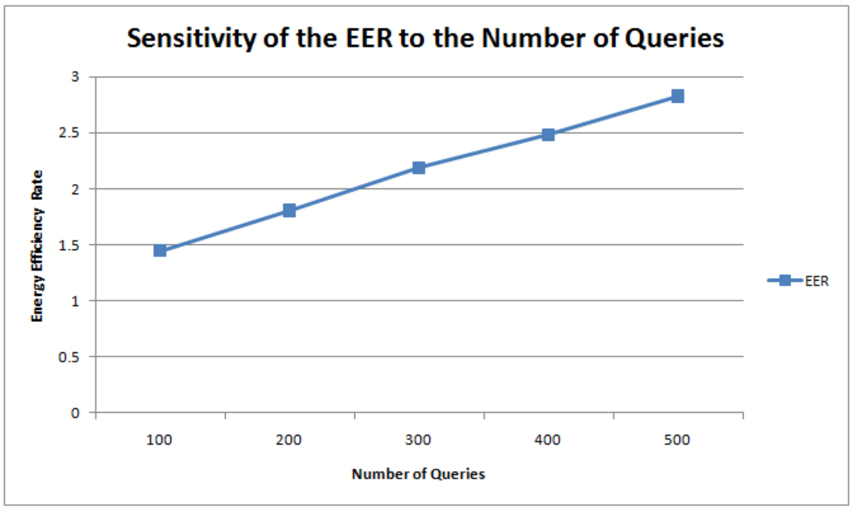
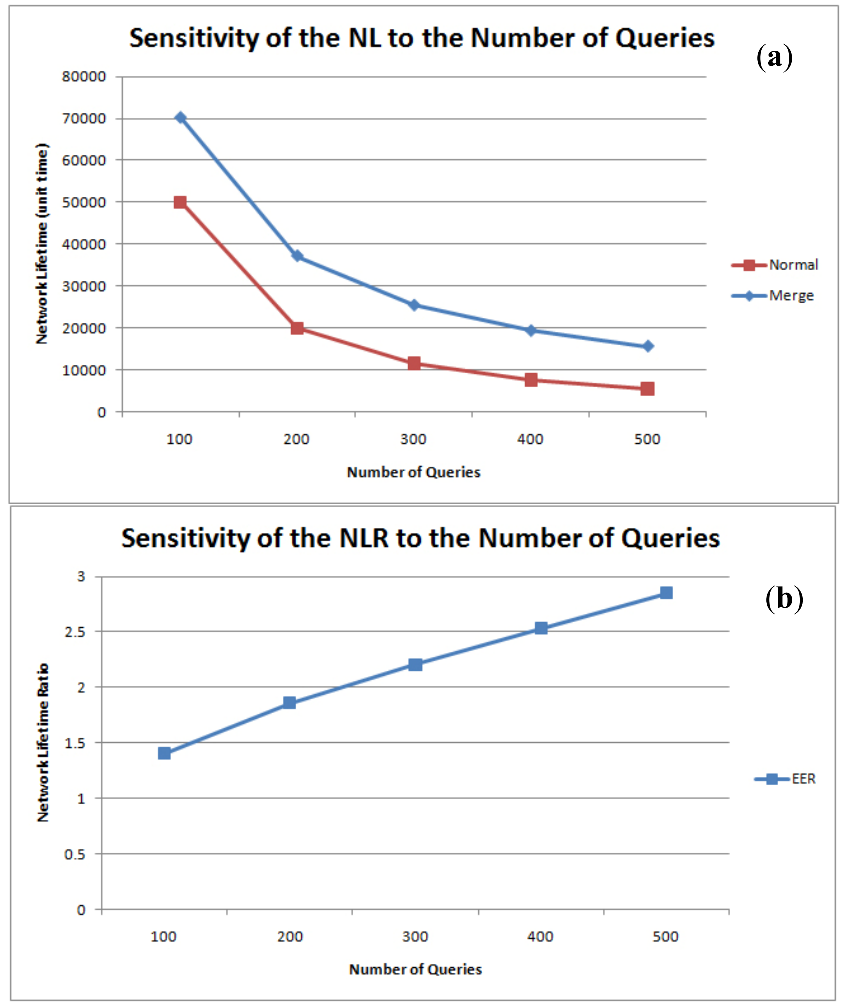


Figure 14(a,b) are the graphs for network lifetime and the NLR against the number of queries. As expected, network lifetime is inversely proportional to the number of queries and the NLR is proportional to the number of total queries.

Figure 14. Sensitivity of the **(a)** Network Lifetime and **(b)** NLR to the number of queries.



6.2.2. Sensitivity to the Period Range

In this simulation, 500 queries were uniformly imposed on a network of 50 nodes with different period ranges. Figure 15 is the graph of the AEC for each period range. Without merging, the AEC is inversely proportional to the period range because greater query response periods indicate less query execution frequencies. Figure 16 shows the energy efficiency rate for each period range. As expected, the query result merging mechanism becomes less efficient as queries are executed less frequently without much overlapping. Therefore, although the proposed mechanism still proves efficient than the normal query processing methods, its efficiency diminishes as query periods are distributed on a larger period scale.

Figure 15. Sensitivity of the AEC to the period range.

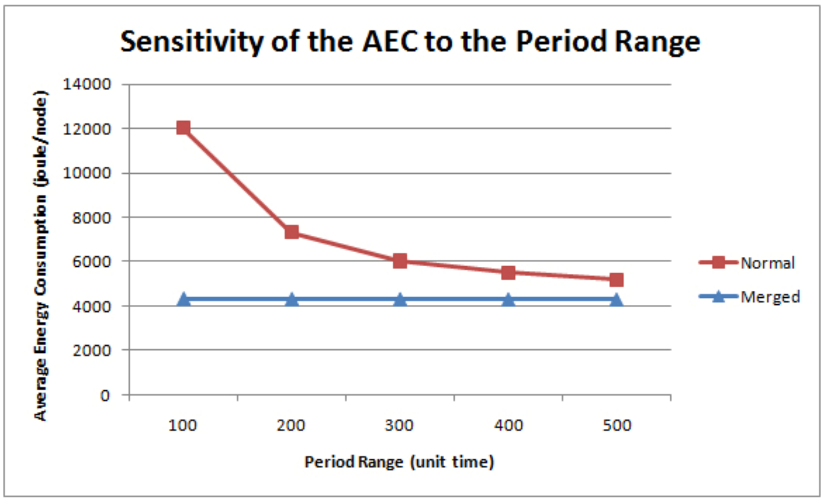


Figure 16. Sensitivity of the EER to the period range.

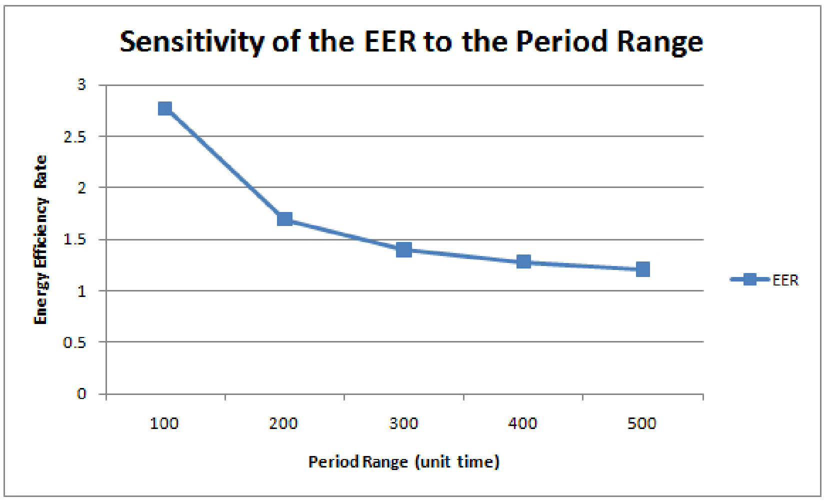
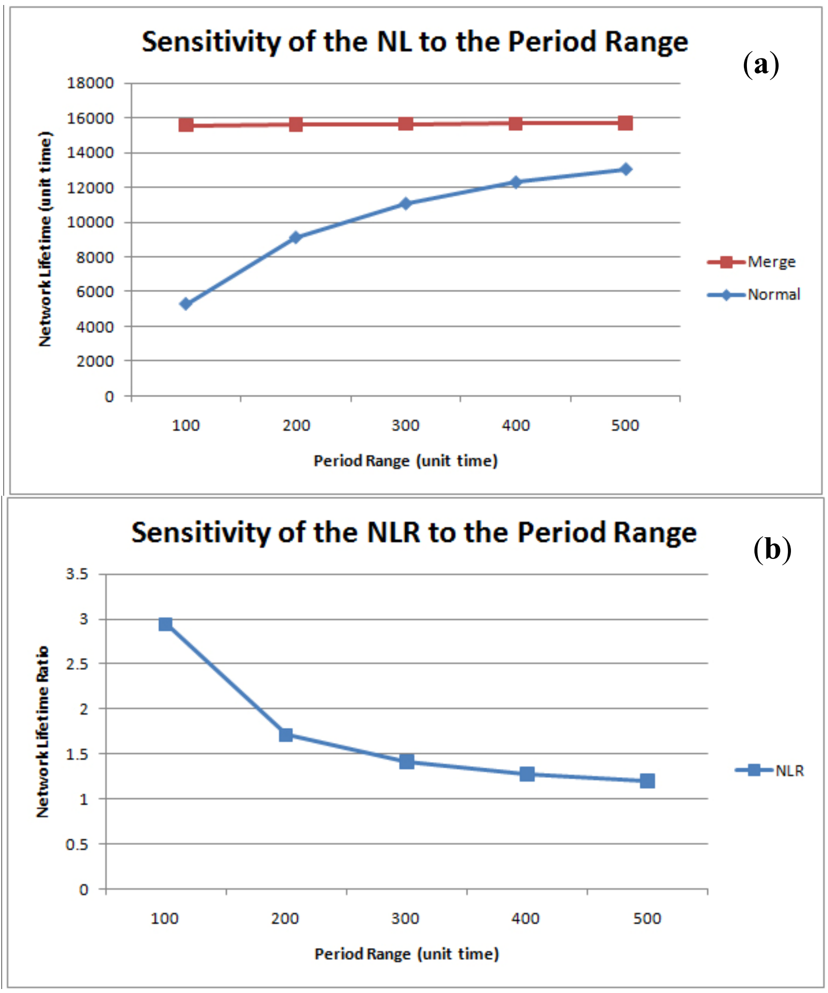


Figure 17(a) shows the sensitivity of the network lifetime to the period range. For a normal network, an increase in the period range decreases the number of query executions during the same time interval, so network lifetime in enhanced as the period range is increased. When query results are merged, however, the network lifetime shows only slight variations since each node always produces one tuple at a time. Hence, the NLR is inversely proportional to the number of queries. Nonetheless, query result merging still proves to be efficient in terms of network lifetime, as shown in Figure 17(b).

Figure 17. Sensitivity of the **(a)** Network Lifetime and **(b)** NLR to the period range.



7. Conclusions

In this paper, we have analyzed theoretically query result merging and proposed a new relational database model view and an efficient query management scheme suitable for underwater sensor networks. We also proposed a message payload format capable of maintaining robustness in the face of unexpected communication failures. Through network simulations, we proved how our query management scheme can efficiently reduce redundant message transmissions and the entailed energy waste, and enhance network lifetime. The management mechanism is expected to display higher efficiency when more queries are generated within a relatively smaller period range.

Furthermore, our contributions are not restricted to underwater sensor networks. The proposed methodology may also be applied to terrestrial RF-based sensor networks to significantly reduce network traffic and node energy waste. With the upsurge in the demand for sensor networks as ubiquitous data computing systems, our query result merging framework has great potentials for providing a systematized sensor network management methodology to multiple users.

It is also worth noting that the proposed scheme doesn't require processing capabilities greater than those of a conventional ubiquitous sensor node. The proposed query models are simple and don't store data history, thus consuming only a limited space of memory. The query condition merging process in the QRP is also a succession of reasonably arduous logical processes manageable with microprocessors of regular capacity, so the proposed query management scheme asks for no more than the processing capability of a regular sensor node.

There are several ways to improve our research results. One significant direction would be moving the focus from underwater sensor networks to RF-based terrestrial sensor networks and assume reliable time synchronization. Researchers have poured great efforts into theoretical studies for time synchronization, but the current technology is too vulnerable to underwater environments for practical time synchronization to happen. But because time synchronization is easy for terrestrial RF-based sensor networks, adjusting our mechanism to time-synchronized sensor networks for terrestrial applications would bring a great enhancement in sensor network query management in general. Also, developing a method for effectively distributing queries or query sets to the target nodes can significantly enhance the performance of our framework. Lastly, increasing our query functionality by adding other relational operations (the "join" operation, for example) would also be meaningful.

Acknowledgements

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2011-C1090-1121-0001) and the research program 2011 of Kookmin University in Korea.

References and Notes

1. Akyildiz, F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless Sensor Networks: A Survey. *Comput. Netw.* **2002**, *38*, 393-422.
2. Romer, K.; Mattern, F. The Design Space of Wireless Sensor Networks. *IEEE Wirel. Commun.* **2004**, *13*, 54-61.
3. Estrin, D.; Culler, D.; Pister, K.; Sukhatme, G. Connecting the Physical World with Pervasive Networks. *IEEE Pervas. Comput.* **2002**, *1*, 59-69.
4. Ian, F.A.; Dario, P.; Tommaso, M. Underwater Acoustic Sensor Networks: Research Challenges. *Ad Hoc Networks* **2005**, *3*, 257-279.
5. Heidemann, J.; Yuan, L.; Syed, A. Research Challenges and Applications for Underwater Sensor Networking. In *Proceedings of IEEE WCNC*, Las Vegas, NV, USA, 3–6 April 2006; pp. 228-235.
6. Freitag, L.; Grund, M.; Singh, S.; Partan, J.; Koski, P.; Ball, K. The WHOI Micro-Modem: An Acoustic Communications and Navigation System for Multiple Platforms. In *Proceedings of IEEE Oceans Conference*, Washington, DC, USA, 17–23 September 2005; pp. 1086-1092.
7. Weiland, M.A.; Deng, Z.D.; Seim, T.A.; LaMarche, B.L.; Choi, E.Y.; Fu, T.; Carlson, T.J.; Thronas, A.I.; Eppard, M.B. A Cabled Acoustic Telemetry System for Detecting and Tracking Juvenile Salmon: Part 1. Engineering Design and Instrumentation. *Sensors* **2011**, *11*, 5645-5660.

8. Deng, Z.D.; Weiland, M.A.; Fu, T.; Seim, T.A.; LaMarche, B.L.; Choi, E.Y.; Carlson, T.J.; Eppard, M.B. A Cabled Acoustic Telemetry System for Detecting and Tracking Juvenile Salmon: Part 2. Three-Dimensional Tracking and Passage Outcomes. *Sensors* **2011**, *11*, 5661-5676.
9. Karl, H.; Willig, A. *Protocols and Architectures for Wireless Sensor Networks*; John Wiley & Sons: Chichester, UK, 2006; p. 21.
10. Syed, A.; Heidemann, J. Time Synchronization for High Latency Acoustic Networks. In *Proceedings of INFOCOM 2006*, Barcelona, Spain, 23–29 April 2006; pp. 1-12.
11. Harris, A., III; Stojanovic, M.; Zorzi, M. When Underwater Acoustic Nodes Should Sleep with One Eye Open: Idle-Time Power Management in Underwater Sensor Networks. In *Proceedings of ACM WUWNet*, Los Angeles, CA, USA, 25 September 2006; pp. 105-108.
12. Xie, P.; Zhou, Z.; Nicolaou, N.; See, A.; Cui, J.H.; Shi, Z. Efficient Vector-Based Forwarding for Underwater Sensor Networks. *EURASIP J. Wirel. Commu. Netw.* **2010**, doi:10.1155/2010/195910.
13. Chirdchoo, N.; Soh, W.; Chua, K.C. Aloha-Based MAC Protocols with Collision Avoidance for Underwater Acoustic Networks. In *Proceedings of the IEEE Conference on Computer Communications*, Anchorage, AK, USA, 6–12 May 2007; pp. 2271-2275.
14. Ong, K.G.; Yang, X.; Mukherjee, N.; Wang, H.; Surender, S.; Grimes, C.A. A Wireless Sensor Network for Long-Term Monitoring of Aquatic Environments: Design and Implementation. *Sensor Lett.* **2004**, *2*, 48-57.
15. Chagas, L.D.; Lima, E.P.; Neto, P.F.R. Real-Time Database Techniques in Wireless Sensor Networks. In *Proceedings of the 2010 Sixth International Conference on Networking and Services*, Cancun, Mexico, 7–13 March 2010; pp. 182-187.
16. Bonnet, P.; Gehrke, J.E.; Seshadri, P. Querying the Physical World. *IEEE Person. Commun.* **2000**, *5*, 10-15.
17. Ozsu, M.T.; Valduriez, P. Distributed Database Systems: Where Are We Now? *Computer* **1991**, *24*, 68-78.
18. Heidemann, J.; Li, Y.; Syed, A.; Wills, J.; Ye, W. *Underwater Sensor Networking: Research Challenges and Potential Applications*; USC/ISI Technical Report ISI-TR-2005-603; Information Sciences Institute, University of Southern California: Marina Del Rey, CA, USA, 2005.
19. Yu, W.; Le, T.N.; Xuan, D.; Zhao, W. Query Aggregation for Providing Efficient Data Services in Sensor Networks. In *Proceedings of the IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, Fort Lauderdale, FL, USA, 25–27 October 2004; pp. 31-40.
20. Bonnet, P.; Gehrke, J.E.; Seshadri, P. Towards Sensor Database Systems. In *Proceedings of the Second International Conference on Mobile Data Management*, Hong Kong, 8–10 January 2001; pp. 3-14.
21. Madden, S.R.; Franklin, M.J.; Hellerstein, J.M.; Hong, W. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Trans. Database Syst.* **2005**, *20*, 122-173.
22. Intanagonwisat, C.; Govindan, R.; Estrin, D. Directed Diffusion: A Scalable and Robust Communication. In *Proceedings of ACM MibileCom'00*, Boston, MA, USA, 6–11 August 2000; pp. 56-67.
23. Krishnamachari, B.; Estrin, D.; Wicker, S. Modelling Data-Centric Routing in Wireless Sensor Networks. In *Proceedings of IEEE INFOCOM '02*, New York, NY, USA, 23–27 June 2002.

24. Chakeres, I.D.; Belding-Royer, E.M. AODV Routing Protocol Implementation Design. In *Proceedings of the International Conference on Distributed Computing Systems Workshops (ICDCSW)*, IEEE Computer Society: Washington, DC, USA, 2004; pp. 698-703.
25. Madden, S.; Franklin, M.J.; Hellerstein, J.M.; Hong, W. TAG: A Tiny Aggregation Service for *Ad-Hoc* Sensor Networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, Boston, MA, USA, 9–11 December 2002; pp. 131-146.
26. Pompili, D.; Melodia, T.; Akyildiz, I.F. Distributed Routing Algorithms for Underwater Acoustic Sensor Networks. *IEEE Trans. Wirel. Commun.* **2010**, *9*, 2934-2944.
27. Erol-Kantarci, M.; Mouftah, H.T.; Oktug, S. A Survey of Architectures and Localization Techniques for Underwater Acoustic Sensor Networks. *IEEE Commun. Surv. Tutor.* **2011**, *3*, 487-502.
28. Meng, M.; Yang, J.; Xu, H.; Jeong, B.S.; Lee, Y.K. Query Aggregation in Wireless Sensor Networks. *Int. J. Multimedia Ubiquit. Eng.* **2008**, *3*, 19-26.
29. Kim, D.H.; Cho, Y.M.; Kim, C.H.; Kim, S.K.; Park, S.H.; Kang, T.W. E-ITRC Protocol with Long & Adjustable Range on Underwater Acoustic Sensor Network. In *Proceedings of Advanced Information Networking and Applications Workshops* (*AINAW '07)*, Niagara Falls, ON, Canada, 21–23 May 2007; pp. 665-672.
30. Chang, Y.I.; Shin, S.Y.; Park, S.H.; Lee, G. Dual Super Cluster Head Underwater Sensor Network Routing Protocol. *J. Korea Soc. Simulat.* **2006**, *15*, 87-96.
31. Ovaliadis, K.; Savage, N.; Kanakaris, V. Energy Efficiency in Underwater Sensor Networks: A Research Review. *J. Eng. Sci. Tech. Rev.* **2010**, *3*, 151-156.
32. Link-Quest, SoundLink Underwater Acoustic Modems. Available online: http://www.link-quest.com/html/uwm_hr.pdf (accessed on 20 August 2011).