Sensors & Transducers, Vol. 189, Issue 6, June 2015, pp. 71-81

Sensors & Transducers

http://www.sensorsportal.com

Comparative Study of Temporal Aspects of Data Collection Using Remote Database vs. Abstract Database in WSN

Abderrahmen BELFKIH, Bruno SADEG, Claude DUVALLET, Laurent AMANTON
University of Le Havre, BP 1123, 76063 Le Havre cedex, France
Tel.: (+33) 2 32 74 43 53, fax: (+33) 2 32 74 43 14
E-mail: Abderrahmen.Belfkih@univ-lehavre.fr, Bruno.Sadeg@univ-lehavre.fr, Claude.Duvallet@univ-lehavre.fr, Laurent.Amanton@univ-lehavre.fr

Received: 4 June 2015 /Accepted: 29 June 2015 /Published: 30 June 2015

Abstract: The real-time aspect is an important factor for Wireless Sensor Network (WSN) applications, which demand a timely data delivery, to reflect the current state of the environment. However, most existing works in the field of sensor networks are only interested in data processing and energy consumption, to improve the lifetime of WSN. Extensive researches have been done on data processing in WSN. Some works have been interested in improving data collection methods, and others have focused on various query processing techniques. In recent years, the query processing using abstract database technology like Cougar and TinyDB has shown efficient results in many researches. This paper presents a study of timing properties through two data processing techniques for WSN. The first is based on a warehousing approach, in which data are collected and stored on a remote database. The second is based on query processing using an abstract database like TinyDB. This study has allowed us to identify some factors which enhance the respect of temporal constraints. *Copyright © 2015 IFSA Publishing, S. L.*

Keywords: Wireless sensor network, Data collection, Query processing, Sensors database, Time constraints.

1. Introduction

Wireless Sensor Networks (WSN) can be considered as a type of ad hoc wireless networks. They are composed of small wireless nodes, which are manually or randomly deployed in a region of interest, to sense different physical characteristics of the environment, like temperature, humidity, pressure, light, and so on. The nodes transmit the sensed data to a sink node referred to as Base Station (BS). This process, called data collection, must be able to meet certain deadlines, i.e. to update data before the expiration of their validity intervals, in order to reflect the current state of the environment.

WSN applications, such as industrial process monitoring and control, can be considered as time critical. They require strict deadlines to send data to the sink nodes. WSN are often deployed without giving some importance to the temporal constraints including the data freshness constraint, and the deadline constraints [27].

Data freshness constraint imposes a limit on the data validity period from the time it is read from a sensor. The constraint of data deadline also requires that data have to be collected within a deadline [2]. In real-time applications, data freshness is a key performance factor. It is important to study the efficiency of data management mechanisms in WSN.

Nowadays, researchers are interested in data processing techniques to offer an efficient solution to increase the WSN lifetime. They have proposed new techniques for data aggregation, data collection and query processing, to simplify the data extraction and management in WSN. The query processing systems such as Cougar [6] and TinyDB [13] use the abstract database approach which considers the WSN as a relational database. They aim to improve the data extraction mechanisms in WSN to save energy and to reduce the communication losses [7].

In this paper, we are interested in investigating two approaches for data processing in WSN:

1) A query processing approach using an abstract database,

2) A periodic data collection using a traditional database.

We simulate temporal constraints in order to identify which method gives better data arrival time to the user and that ensures data freshness in WSN. We consider the following performance parameters:

1) The network convergence time, which is the time that a sensor node takes to connect to the base station, to build and to update its routing tables. It can affect the arrival time of data,

2) The data collection time,

3) The database response time.

The data freshness can be improved, once these parameters are faster.

The remainder of this paper is structured as follows. Section 2 reviews related work. In Section 3, we describe the two approaches cited previously. In Section 4, we carry out simulations, and we present and comment the results obtained. Finally, in Section 5, we conclude the paper by showing how to effectively exploit the two techniques and we give some perspectives to this work.

2. Related Works

In this section, we give an overview of some research works, which have studied the time criterion for WSN data processing. Thereafter, we present different abstract databases for sensor networks.

2.1. Data Freshness in WSN

The sensor data must be fresh to reflect correctly the state of the environment, but they have also to be temporarily correlated [29]. Many works have studied data freshness in WSN, among which we cite the following work:

Suriyachai, *et al.* have discussed the time-critical data delivery in WSN [16]. They have presented a novel TDMA[1] based Medium Access Control (MAC) protocol named GinMAC, which incorporates routing, topology control and reliability control mechanisms. It aims to ensure timely data delivery and the reliability control mechanism, with minimum energy consumption in WSN. The authors have shown that GinMAC is able to balance delay, reliability and energy consumption requirements for the system with time-critical data delivery in WSN.

Sharaf, *et al.* have presented a new approach based on SQL-Like query syntax, named TiNA (Temporal Coherecy-Aware in-Network Aggregation) [15]. TiNA exploits temporal coherency tolerance, to reduce the number of transmitted messages by each node. The authors have proposed a new clause named TCT (Temporal Coherency Tolerance) for each query, in order to express temporal coherency tolerance. For example, query Q1 bellow involves getting the total light for rooms and grouping by floor, with TCT = 10 %. The new reading is not reported if the difference between it and old one is smaller than the associated TCT value.

Q1: **SELECT** {FLOOR, SUM (LIGHT)}
FROM SENSORS
GROUP BY {FLOOR}
EPOCH DURATION 30 s
VALUES WITHIN 10 %

Choi, *et al.* [4] consider that the aggregation time in WSN can be affected by the deadline constraints, i.e., data validity period or data delivery deadline. They have proposed a new aggregation time control (ATC) algorithm, to reduce energy consumption in the network and to adjust the aggregation time of the node to ensure data delivery without exceeding fixed deadlines.

Hiromori, *et al.* have proposed a new protocol, named DD (Drainage Divide), for periodic data collection in WSN multi-sinks [9]. DD transforms the network into a set of trees, rooted at the sinks. Then, it adjusts them, so that the delay from each node to the corresponding sink meets given deadlines. The authors have also applied the CSMA/CA-based MAC delay analysis to estimate the transmission delay at each node. DD protocol has guaranteed data delivery within time, and it has satisfied the given delay constraints in WSN.

There are many research efforts to study temporal constraints such as delay constraints, deadline constraints and data delivery deadline on data processing techniques like data aggregation, data collection and query processing. However, there are few existing research works which focus on studying temporal constraints in query processing systems, also called abstract databases.

2.2 Abstract Database for WSN

Abstract databases for WSN have been cited for the first time by Bonnet, *et al.* [3]. The authors consider the WSN as a relational database that can be queried using SQL-like queries. Abstract databases present a declarative approach to facilitate the

[1] Time Division Multiple Access

description and processing of queries in the WSNs. Data required by the base station present a virtual table, in which columns represent the data requested by the user.

Madden, *et al.* have presented a new query processing system, named TinyDB, to extract information from a network and using TinyOS [12] as an operating system. TinyDB is composed of two subsystems:

1) TinyDB application, running on each node of the network,

2) Java-based client interface, allowing the user to describe the data using a declarative SQL-LIKE queries, i.e., without knowing how the data are processed in the network. The users write the query that will be compiled and split into tasks, then injected into the network. Once a node receives the query, it processes, and returns the results to the user via the base station. For example, to calculate the average temperature for all sensors when the light is greater than 400 for a period of 10 seconds, we write the following query Q2:

Q2: **SELECT** AVG (temperature)
FROM sensors
WHERE light > 400
EPOCH DURATION 10 s

Another database abstraction is TikiriDB [11] which is a layer for ContikiOS [26]. It provides functionality to collect data from WSN without having any information about programming knowledge of sensor nodes. TikiriDB provides a TikiriSQL library, to receive queries from client, to parse them and to generate query packets. The results are sent to the gateway computer via the Serial Forwarder (SF) application. The node connected to the gateway, receives the queries, then it injects them in WSN. It receives thereafter the results, which will be forwarded to the user the serial forwarder application. Q3 is an example of TikiriDB query:

Q3: **SELECT** light, temperature
FROM sensors
WHERE temperature > 50
SAMPLE PERIOD 2 s FOR 10 s

Query Q3 permits to get the light and the temperature from all sensors whose temperature is above 50. Each new reading will be delivered to the user every 2 seconds for duration of 10 seconds.

Amato, *et al.* have proposed MaD-WiSe (Management of Data in Wireless Sensor networks) [1], which considers a WSN as a highly distributed and dynamic database. MaD-WiSe takes account of various aspects related to database system design. Users can express queries using an SQL-like language, named MW-SQL, to collect, to filter, and to manage sensors data. MW-SQL uses the concept of the source of data (sensor id, room, etc.) to specify the data that will be retrieved from the network. It also includes temporal aggregates "EVERY" and "EPOCH" clauses. "EVERY" clause is used to specify the periodic sensor data acquiring rate and the "EPOCH" clause is used to define the time interval of data collection from WSN. Q4 is an example of MaD-WiSe query:

Q4: **SELECT** min (1.Light), avg (2.Temperature)
FROM 1.Light, 2.Temperature
WHERE 2.Temperature > 20
EPOCH 10 SAMPLES
EVERY 30000

In Q4, the user requests the minimum light from sensor with id=1 and the temperature average from sensor with id=2 whenever the corresponding temperature readings exceed 20°. Data will be taken every 30 seconds and aggregation will be performed every 10 samples (i.e., every 20 seconds).

Corona [10] is a distributed in-network query processing system which provides a declarative query SQL-Like language to formulate queries. Corona introduces the notion of freshness into WSN, allowing the user to obtain data from a sensor network with data freshness guarantees. It uses multiple aggregate queries to reduce processing delays and costly communication in WSN. It is able of running multiple applications simultaneously. Q5 is an example of Corona query.

Q5: **SELECT** temperature
FROM sensors
WHERE temperature > 50
EPOCH 60 s
FRESHNESS 10 s

In Q5, the user queries all sensors to get a temperature greater than 50 for an epoch period of 60 seconds. It defines also the freshness constraint clause of 10 seconds to fix the time allowed between two acquisitions.

Amol, *et al.* proposed a data acquisition model named "BBQ" [20], based on probabilistic modeling techniques to optimize data acquisition from sensor network. This model is based on time varying multivariate Gaussians. User sends a SQL query translated into probabilistic computations over the model including specific parameters like error bound indicating allowing the user to specify the approximation value tolerance and confidence threshold. Q6 is an example of probabilistic query used in BBQ model:

Q6: **SELECT** nodeid, temp ± 0.1 °C,
Confidence (0.95)
FROM Temperature AS T
WHERE T.temp > 30 °C

In Q6, the user requests the node ID and the temperature values with an error bound more or less 0.1 °C *i.e.* the difference between the real value of temperature and the value detected, must not exceed 0.1 °C, also with a confidence equal to 95 %, and

greater than 30 °C. Based on the model and sensor readings values, the system decides the efficient way to answer the query with the requested confidence. BBQ model is updated over the time, based on new reading values which are reported as a probability density function. Thereafter, BBQ generates an observation plan and sends it over the network in order to select observations that minimize cost and maximize confidence. We note that temporal constraints have not been considered in the most of abstract databases. They have been studied only in the recent Corona work [10]. In the next section, we illustrate our network model through two scenarios: a periodic data collection and a query processing using TinyDB and we present the parameters we selected to discuss them in Section 4.

3. Network Model

In the two scenarios, we have used a network model composed of four components:

1) Sensor nodes to report information about the environment,

2) A base station to collect periodically data from sensor nodes,

3) A remote database used to store the received data,

4) A user which can connect to the database and get information about WSN.

In the second scenario, we have added an abstract database for WSN, named TinyDB, which allows the user to send queries to the base station and to get answers.

3.1. First Scenario: Data Collection with Remote Database

In this scenario (Fig. 1), sensor nodes are randomly deployed over an area and the base station is placed outside the network. Nodes communicate amongst themselves through wireless communications. Sensor nodes retrieve data values like temperature and humidity, and periodically send them to a base station. The data received are transmitted to the remote database to be stored. Finally, they can be displayed through an interface to the users. The remote database has been deployed in order to manage and storage the data collected from WSN. The user can query the database using SQL-like queries to get information about the WSN.

3.2. Second Scenario: Query Processing with WSN Abstract Database

This scenario is based on a query processing technique, in which we use an abstract database system, TinyDB (Fig. 2). User specifies the data that will be retrieved from WSN, through SQL-like declarative queries. It sends them to the base station via the abstract database interface. The base station will inject these queries over the network. Each sensor receives a query, responds with the values requested for a duration fixed in the query by the user. The base station sends the responses to the user via the abstract database system and they can be stored in a remote database. We have considered some performance parameters for the two approaches and we discussed them in next section.

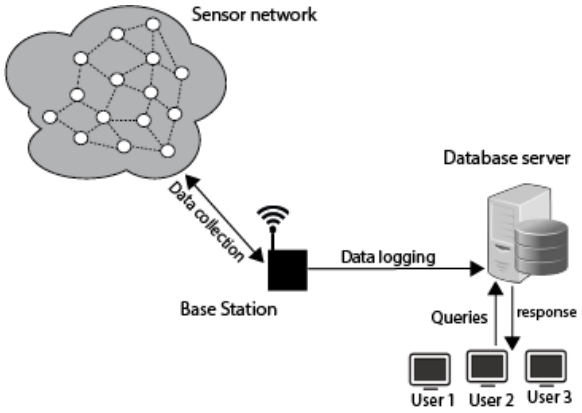


Fig. 1. Data collection with remote database.

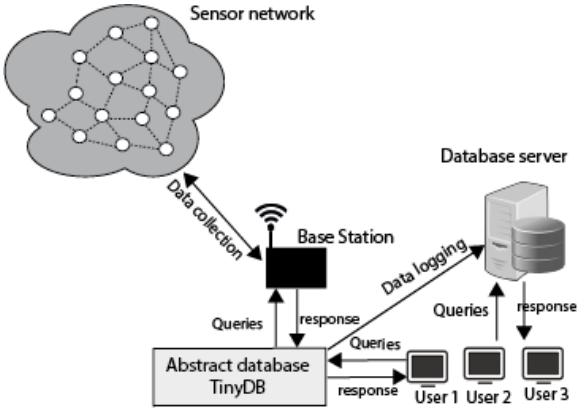


Fig. 2. Query Processing with WSN Abstract Database.

These parameters are:

1) The network convergence time, an important factor to ensure the temporal constraints in WSN. It precedes the processing of data collection. It can affect the arrival time of data when it takes a long time. We consider also

2) The data collection time, which represents an important factor to guarantee the freshness of data and the data delivery deadline. When the data collection is fast, data freshness will be better,

3) The database response time, which can be considered as an important factor in the first scenario, because it represents the only data available to the user.

In the next section, we implement the two scenarios described above and we discuss the

performance parameters for the two approaches. We identify and we analyze also different impacts which can affect these parameters.

4. Simulation and Results

4.1. Simulation Environment

4.1.1. Data Collection Scenario

We have used COOJA [14], the network simulator for ContikiOS [19], to create the first scenario. We have created one sink node as base station and a set of sender nodes of type Tmote Sky with 48 KB of flash memory. The nodes are randomly distributed in an area within 100 m×100 m and the sink is placed outside of the deployment area. The number of sensors varies from 10 to 100 with step of 10. Each node sends temperature and humidity values to the sink node every 60 seconds. Table 1 summarizes the simulation parameters.

Table 1. Simulation Parameters.

Parameters	**Values**
Operating system	ContikiOS 2.6
Simulator	Cooja
Mote type	Tmote Sky
Simulation area (m)	100×100
Number of nodes	10-100
Number of sink Nodes	1
Source data rate	Random
Radio range (m)	80
Data packet size (bytes)	15

We use RPL (Routing Protocol for Low-Power and Lossy Networks) [18] to provide communication between sensor nodes and the base station. RPL is a routing protocol designed for low power and lossy networks and for large networks. It is more efficient for data delivery time than other existing known protocols such as LOAD (LOAD is derived from AODV), DSR (Dynamic Source Routing) and DSDV (Destination-Sequenced Distance Vector routing) [17]. It provides efficient routing paths guaranteeing data delivery before deadline in WSN.

We have used ContikiMac radio duty cycling protocol [5] inspired from existing duty cycling protocols. It has shown efficient time management because it uses a wake-up mechanism to define a precise timing between packet transmissions. We have used the default ContikiMac layer CSMA (carrier sense multiple access). Table 2 below summarizes the different protocol layers used in simulation.

Table 2. Simulation layer protocol parameters.

Transport layer	UDP
Network layer	IPV6/RPL
Radio duty cycling	ContikiMac
MAC layer	CSMA
Physical layer	IEEE 802.15.4

4.1.2. Query Processing Scenario

We used TinyDB as abstract database in TinyOS. TinyDB runs on Mica2 motes with 128 kB flash memory because its code execution requires a flash memory upper than 48 kB. We have chosen to use TinyDB because it is the most widely used system by researchers and it has shown its effectiveness in maintaining energy and for data management. The network is based on Mica2 sensors randomly scattered in the field. We have varied the number of sensors from 10 to 100 by step of 10. The sensor is queried on demand by the user via the base station. Table 3 summarizes the simulation parameters.

Table 3. Simulation Parameters.

Parameters	**Values**
Operating system	TinyOS 1.x
Simulator	Tossim
Mote type	Mica 2
Simulation area (m)	100×100
Number of nodes	10-100
Number of sink Nodes	1
Radio range (m)	80
Data packet size (bytes)	20

TinyDB uses a tree-Based Routing Protocol [8] to provide the query dissemination and the result collection. Each node forwards the query to other nodes in the network, to form the routing tree. This process ends when all nodes have received the query. TinyDB uses CSMA protocol, which is the already existing default MAC protocol used by TOSSIM simulator. It uses also the Low Power Listening (LPL) for low power communication and the Drip/Drain routing. Drip is a transport-layer component for disseminating messages throughout a network and Drain is a collection routing layer for TinyOS 1.x. Table 4 summarizes the different protocol layers used in simulation.

Table 4. Simulation layer protocol parameters.

Transport layer	Drip/Drain
Network layer	Tree-based routing protocols
Radio duty cycling	Low-Power Listening
MAC layer	CSMA
Physical layer	IEEE 802.15.4

4.2. Simulation Description

4.2.1. Data Collection Scenario

Each sender node uses the UDP sender algorithm to send its data to the base station according to a transmission period. We have defined a transmission period equal to 60 seconds. We have chosen a random waiting time (SEND TIME), before sending packets to avoid that the senders sending packets at the same time. Thereafter, we have activated the digital SHT11 of sensor, which is relative to sensor Humidity and temperature. Sender node listens to its neighbors using TCP/IP handler function and continues to send data currently detected to the base station using the "send packet" function. The base station receives data from a sensor network, then it stores them in remote database using INSERT queries. Algorithm 1 presents the data collection process included in each sensor.

Algorithm 1: UDP sender process

```
1:  define period 60
2:  define send time random Value
3:  sensor activate (sht11 sensor)
4:  while 1 do
5:     if ev = TCP/IP event then
6:          TCP/IP handler()
7:     end if
8:     if etimer expired(period) then
9:          Etimer reset (period)
10:         Ctimer set (backoff time, send time,
11:         send packet, null)
12:    end if
13: end while
```

4.2.2. Query Processing Scenario

When TinyDB receives the query from the user, it parses and translates it to SQL query into an execution plan. Before injecting the query in the network, TinyDB uses a method query named Semantic Routing Trees (SRT) [8] to determine which node in the routing tree will participate and has capabilities to produce query results.

SRT is composed of two phases: the first phase consists in building initial tree formation allowing parent nodes to know about the capabilities of its children. SRT disseminates a query including the type of information requested. Each node selects its parent and responds with requested information. Then, each parent receives the id and range of values from its children. In the same way, each parent chooses its own parent until the root node has information from all of its children. In the second phase, the real query is disseminated to nodes having relevant responses.

Once queries have been disseminated. Each node receives a query, and begins a query execution which includes filtering and aggregation of results according to the query plan, then data are delivered to parent nodes and thereafter to the base station. TinyDB adjusts result transmission rates to reduce power consumption. When a result arrives, TinyDB calls the result listener method and displays the query result for the user using a Java-based client interface.

4.3. Simulation Results

4.3.1. Impact of Number of Nodes on Network Convergence Time

The network convergence time is defined as the time needed for the sensors to connect to the base station and to build routing tables. This step comes just before the process of data collection. It can affect the data arrival time. The shorter is the convergence time. The quicker is the availability of the data. In Fig. 3, the network convergence time increases when the number of nodes increases. The network convergence time depends on the DODAG building process, which begins at the root (base station). The root starts the dissemination of information about the structure using DIO (DODAG Information Object) messages.

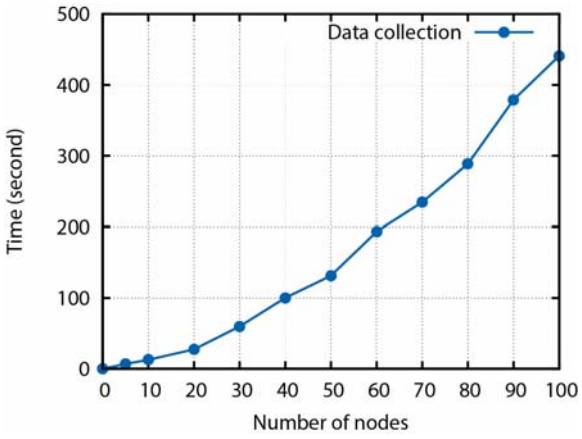


Fig. 3. Network convergence time for data collection.

The nodes, which are not connected to the tree (without communication with the root) receive the message DIO and process it. Then they decide to join or not the structure. Once the node has joined the structure, it has a route to the root of the DODAG structure. Building the final routing table depends on the number of nodes and the path cost between nodes. We note also in Fig. 4 that the convergence time for TinyDB (acquisition on demand) is less than that of data collection (periodically).

TinyDB constitutes a routing tree or Directed Acyclic Graph (DAG) with all sensors in which the root is the sink (gateway).

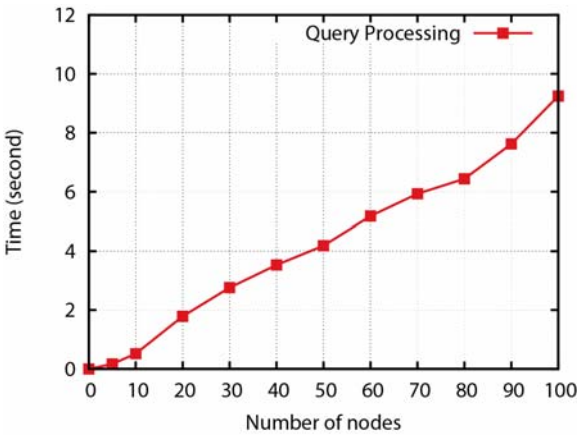


Fig. 4. Network convergence time for TinyDB.

TinyDB uses a selecting multi-hop tree based on efficiency of routing which depends on energy consumption. Sensors make intelligent routing decisions, where every node selects a parent based on link quality to the base station. The nodes keep both a short list of neighbors which have a recently heard transmission, and some routing information about the connectivity of those neighbors to the rest of the network. They associate also a link quality with each of their neighbors.

4.3.2. Impact of Number of Nodes on Data Collection Time

Data collection time represents the time taken to get the responses from all sensors connected to the base station. It can affect the data validity time and then not reflecting the current state of the environment. The time required to send the data from sensor to the base station depends on the sensors capacity, its position, while the path quality depends on the number of hops to the base station. Generally, the sender uses multi-hop paths to send its data when the base station is far located, which leads to transmission delay and can cause a failure, i.e. to not respect the data validity time.

Fig. 5 shows the time spent by the nodes to complete the sending of their data, using both data collection technique and query processing technique.

We observe that the curve increases when the number of sensors increases for the two techniques. The time required to send the data depends of the number of hops and the availability of parent nodes to send data received from children. Their availability is not guaranteed all time because the choice of parent node depends on the node decision, which consists on joining or not the structure.

We also notice in Fig. 5 that the time required to obtain results from all sensors by using the technique of processing request is less than that of data collection. With TinyDB, the data is regularly reported and aggregated by a tree or a directed acyclic graph from the nodes to an access point network. It includes aggregation and filtering operations inside the sensor network to maintain all routing information. The parent nodes near the root put agreements with their children on a time interval to receive data from them. The whole process is repeated for each period and query.

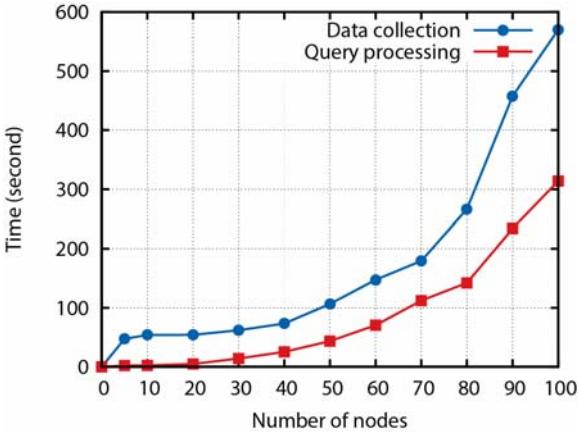


Fig. 5. Completed Cycle Time.

4.3.3. Impact of the MAC Layer Protocols

The goal of this test is to observe the Complete Cycle Time (CCT) vs. Duplicate Packets Sending (DPS) with either a periodic data collection, or a query processing system (TinyDB). We have varied the number of sensors in each time.

We note in Table 5 that the number of transmission packets with TinyDB is greater than that with periodic data collection. The MAC-layer protocol can be considered as one of the causes that explains these results.

Table 5. Duplicate sends with periodic data collection.

	Periodic data collection		**Query processing (TinyDB)**	
NB. nodes	**CCT(sec)**	**DPS**	**CCT(sec)**	**DPS**
5	47.430	0	2.203	0
10	53.955	0	2.434	0
20	54.157	0	4.766	21
30	62.082	0	14.060	102
40	73.302	0	25.441	121
50	106.609	0	435.43	174
60	147.357	2	705.02	294
70	179.478	1	112.261	308
80	266.357	15	142.114	405
90	457.570	78	233.737	420
100	570.100	72	313.832	633

TOSSIM implements CSMA based MAC protocol by default. The basic principles of this protocol are "listen before talk and contention". Each node listens on the channel and checks if it is clear before each sending. If the channel is not clear, it waits for a randomly chosen period of time called the back-off, until the channel becomes clear. When the number of sensor nodes increases, the nodes compete for the channel and make the channel occupied. This competition causes a transmission delays and a back off time accumulation, which increases the time needed to collect data from all sensors. On the other hand, each sender repeatedly sends its packet, during the full wake-up, until it receives a link layer acknowledgment from the receiver (sink node), meaning that the receiver has correctly received the packet.

In a periodic data collection, the sky motes implement a radio duty cycling protocol, named ContikiMac using a wake-up mechanism to listen for packet transmissions from neighbors and to keep the transceiver turned off, in order to reduce power consumption in WSN [5]. ContikiMac timing includes the interval between each packet transmission, the interval between each CCA (Clear Channel Assessment), the time between receiving a packet and sending the acknowledgment packet, the time required for successfully detecting an acknowledgment from the receiver and the time required for a stable RSSI (Received Signal Strength Indicator), needed for a stable CCA indication. ContikiMac applies for a set of timing constraints to precise timing between transmissions which explains the decrease in duplicate packets sending.

4.3.4. Impact of Choosing the Database on Average Response Time

This experience consists to collect data from WSN and insert them in a remote database. We use for this purpose three DBMS PostgreSQL, MySQL, and SQLite to evaluate queries response time which represents the total time which the query takes to complete its execution.

PostgreSQL [22] is an object-relational database management system (ORDBMS), based on an open source research project at the University of California called POSTGRES. It supports a large part of the SQL standard and offers many advanced features as transactional integrity, complex queries support, table inheritance. New data types, functions, operators, aggregate functions, and index methods may be added by the user.

MySQL [21] is an Open Source SQL database management system developed, distributed, and supported by Oracle Corporation. It can support foreign key, multi-threaded, partial stored procedures and transactional and non-transactional storage engine. It supports several storage engines that act as handlers for different table types: MyISAM is the default storage engine of MySQL, InnoDB storage has been designed for maximum performance and it supports foreign key referential-integrity constraints.

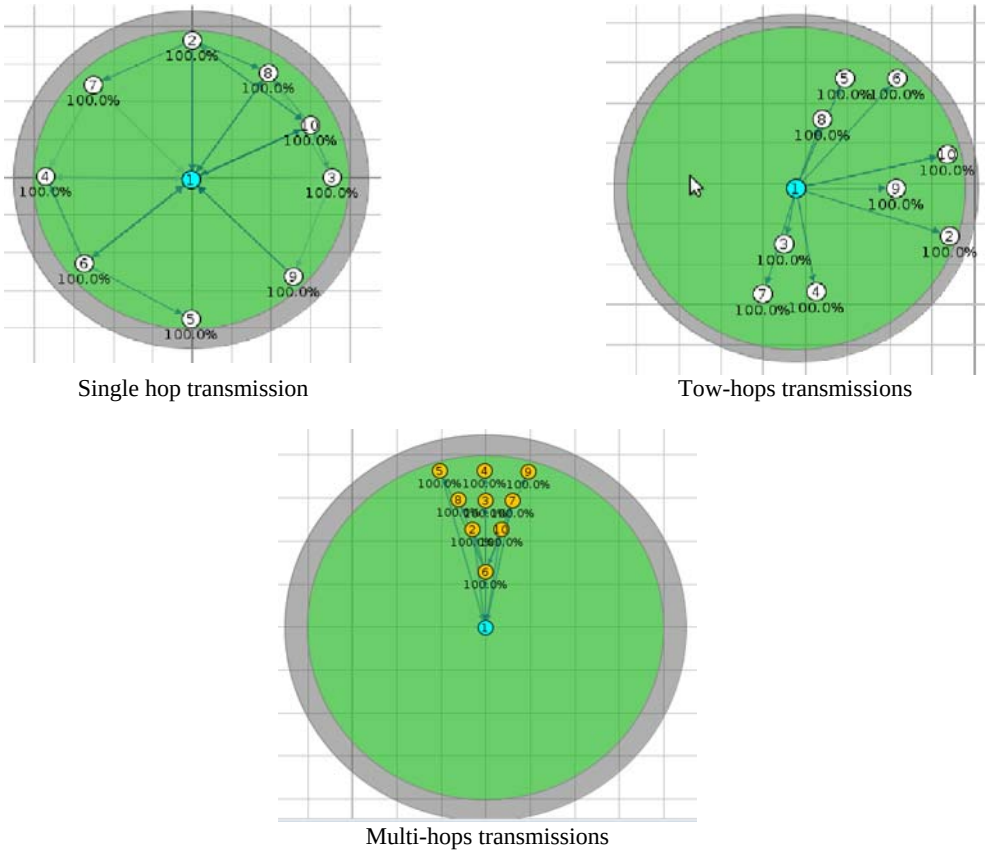
SQLite [23] is a relational database management system and represents a simply scaled-down version of MySQL, PostgreSQL, and other popular database systems. It provides an embedded SQL database engine and it reads and writes directly to ordinary disk files. SQLite does not require server means and setup, it can support tables, indexes, triggers, views and the most of the query language features found in the standard SQL. SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads. Table 6 shows that SQLite database gives the best average response time for the SELECT queries, then MySQL and finally PostgreSQL. For the INSERT queries, PostgreSQL database has the less average response time compared to MySQL and SQLite. SQLite takes a lot of time to insert data, because it does not have a central server to coordinate accesses. It must close and re-open the database file, and invalidate its cache, for each transaction, which increases the response time. SQLite use a locking mechanism, when a write operation happens the entire database must literally be locked, written to, and then unlocked. It does not allow multiple concurrent writes. Whereas, PostgreSQL allows multiple transactions to proceed with inserts concurrently. MySQL is better than PostgreSQL for readings tables because it has a "Query Cache" to make queries faster. Also, it uses InnoDB table that makes users searches more efficient. SQLite is faster than MySQL because it uses a disk access to read and to write directly from the database files on disk. It does not need to make a connection to server.

Table 6. Query response time (ms).

Query type	PostgreSQL	MySQL	SQLite
Insert queries	9.397	48.626	72.788
Select queries	0.992	0.690	0.225

4.3.5. Impact of the Nodes Positions and the Number of Hops on Data Collection Time

In this section, we show the performances of network by changing the position of the nodes. In all experiences, the sink node is 40 meters from each node (Fig. 6). The first experience is to place the sink in the center surrounded by 9 nodes. Each node sends data through single-hop transmission. The second experiment consists of groups of nodes, in which the closest nodes to the base station are selected as parents. Sensors nodes send their data through parent nodes, by making two hops. In the last experience, we have placed the nodes in the form of a tree and observed if this form will improve the data arrival time. All nodes use multi-hop transmissions to send their data.



Single hop transmission

Tow-hops transmissions

Multi-hops transmissions

Fig. 6. Nodes positions and hops transmission.

We can see from Table 7 that the completed cycle time result changes from simulation to another. Usually, the position of the sink and the position of nodes affect the data collection time. We can explain it by the number of hops used to forward data to the sink: higher is the number of hops, longer is the time for data collection.

Table 7. Impact of number of hops on data collection time.

Hop transmission	Completed cycle time (ms)	Average response time (ms)
Single-hop transmission	32.931	3.659
Two-Hop transmission	34.740	3.860
Multi-Hop transmission	53.528	5.947

4.3.6. Impact of Network Topologies on Data Collection Time

Wireless Sensor Network includes different topologies which have a greatly impact on the data collection time. In this experience, we have tested 15 sender nodes and one sink node with four network topologies (Tree, Star, Mesh, Grid).

In tree-based topology, nodes are organized by hierarchy levels, composed by children nodes, parent nodes and relay nodes, which communicate together and transmit their data to a root node (base station) via multi-hop routes.

In star-based topology, the base station is in the center of the network and the other nodes are in the one-hop neighborhood of the base station. In this topology, all nodes communicate via the base station [25].

In mesh-based topology, one node communicates with multiple neighbors and the message can take several paths to achieve the destination. Also, when sensor node failed, the network reorganizes itself to maintain the communication with the base station.

In grid-based topology, the network is divided into square grid of size divided by user. One node is elected as the coordinator node of grid and the responsible of forwarding information and transmitting data packets.

Nodes periodically send to the coordinator data which will be transmitted thereafter to the base station. The result in Table 8 shows the time spent to collect data from all sensors for the two technologies. We note that the tree network topology for data collection technology is faster than the other topologies. In fact, RPL forms a tree-like topology rooted at the sink, reflecting the above results. Star, mesh and grid topologies have not shown good results. RPL takes a time to define DODAG

networks and to build path to the root. It cannot fully exploit the network, which affects the quality of paths.

Table 8. Complete cycle time (ms).

Topologies	Data collection	Query processing
Star	62.312	1.832
Mesh	56.002	1 .362
Grid	54.121	2.163
Tree	53.515	2.143

For the query processing technology, the mesh topology gives the best complete cycle time versus the other topologies. In mesh topology, all nodes cooperate in the distribution of data in the network. The same technology is used by TinyDB. The nodes make intelligent routing decisions, where every node selects a parent based on link quality to the base station.

5. Conclusions

In this work, we have studied and compared the timing properties of the data collection from a sensor network using a static database versus an abstract in-network database, named TinyDB. We have evaluated temporal constraints such as data collection time, database response time, and network convergence time in both approaches. We have found, according to the tests performed, that many factors can affect the temporal constraints in a WSN. We have determined that the network topology and the routing protocol together may play an important role on data collection time. The convergence time also has an impact on the process of data collection. We have shown clearly the timing-response advantage of using a TinyDB approach compared to accessing the data stored in an external database. So, we can conclude that the great choice of the network topology and the routing protocol with the right approach can improve the temporal constraints in WSN. We plan to work in this direction in a future works by taking into account data temporal consistency.

References

[1]. Amato G., Chessa S., Vairo C., Mad-wise: A Distributed Stream Management System for Wireless Sensor Networks, *Software: Practice and Experience*, 40, 5, 2010, pp. 431–451.

[2]. Bhattacharya S., Achieving Application Quality of Service in Resource-constrained Wireless Sensor Networks, PhD Thesis, St. Louis, MO, USA, 2008.

[3]. Bonnet P., Gehrke J., Seshadri P., Towards Sensor Database Systems, in *Proceedings of the International Conference on Mobile Data Management*, 2001, pp. 3–14.

[4]. Choi J. Y., Lee J., Lee K., Choi S., Kwon W. H., Park H. S., Aggregation Time Control Algorithm for Time Constrained Data Delivery in Wireless Sensor Networks, in *Proceedings of the IEEE Vehicular Technology Conference*, Melbourne, 2006, pp. 563-567.

[5]. Dunkels A., The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, *Swedish Institute of Computer Science*, 2011.

[6]. Fung W. F., Sun D., Gehrke J., Cougar: the Network is the Database, in *Proceedings of the International Conference on Management of Data (ACM)*, New York, USA, 2002, pp. 621.

[7]. García-Hernando A.-B., Martínez-Ortega J.-F., López-Navarro J.-M., Prayati A., Redondo-López L., Problem Solving for Wireless Sensor Networks, Computer Communications and Networks, *Springer*, 2008.

[8]. Gehrke J., Madden S., Query Processing in Sensor Networks, *IEEE Pervasive Computing*, 2004, pp. 46–55.

[9]. Hiromori A., Uchiyama A., Yamaguchi H., Higashino T., Deadline-Aware Data Collection in Csma/Ca-Based Multi-Sink Wireless Sensor Networks, in *Proceedings of the International Conference on Mobile Computing and Ubiquitous Networking*, 2012, pp. 1–7.

[10]. Khoury R., Dawborn T., Gafurov B., Pink G., Tse E., Tse Q., Almi'Ani K., Gaber M. M., Rohm U., Scholz B., Corona: Energy-Efficient Multiquery Processing in Wireless Sensor Networks, *Database Systems for Advanced Applications (DASFAA)*, 2010, pp. 416–419.

[11]. Laxaman N., Goonatillake M., De Zoysa K., TIKIRIDB: Shared Wireless Sensor Network Database for Multi-User Data Access, *CSSL*, 2010.

[12]. Levis P., Madden S., Polastre J., Szewczyk R., Whitehouse K., Woo A., Gay D., Hill J., Welsh M., Brewer E., Culler D., TinyOS: An Open Operating System for Wireless Sensor Networks, *Ambient Intelligence*, Vol. 35, 2005, pp. 115–148.

[13]. Madden S., J. M. Franklin, M. J. Hellerstein, Hong W., The Design of an Acquisition Query Processor for Sensor Networks, in *Proceedings of the International Conference on Management of Data*, 2003, pp. 491-502.

[14]. Osterlind F., Dunkels A., Eriksson J., Finne N., Voigt T., Cross-level Sensor Network Simulation with COOJA, in *Proceedings of the Annual IEEE Conference on Local Computer Networks*, Tampa, Florida, USA, 2006, pp. 641–648.

[15]. Sharaf M. A., Beaver J., Labrinidis A., Chrysanthis P. K., TiNA: A Scheme for Temporal Coherency Aware In-Network Aggregation, in *Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access*, New York, USA, 2003, pp. 69–76.

[16]. Suriyachai P., Brown J., Roedig U., Time Critical Data Delivery in Wireless Sensor Networks, *Distributed Computing in Sensor Systems, Lecture Notes in Computer Science*, Vol. 6131, 2010, pp. 216-229.

[17]. Vucinic M., Tourancheau B., Duda A., Performance Comparison of the RPL and Loadng Routing Protocols in a Home Automation Scenario, in *Proceedings of the Wireless Communications and*

Networking Conference (WCNC), 2013, pp. 1974–1979.
[18]. Winter T., Thubert P., Brandt A., Hui J., Kelsey R., Levis P., Pister K., Struik R., Vasseur J., Alexander R., RPL: IPV6 Routing Protocol for Low-Power and Lossy Networks, in *Proceedings of the Seminar of Network Architectures and Services*, 2012, pp. 59-66.
[19]. Dunkels A., Gronvall B., Voigt T., Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors, in *Proceedings of the Annual IEEE Conference on Local Computer Networks (LCN'04)*, 2004, pp. 455-462.
[20]. Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, Wei Hong, Model-Driven Data Acquisition in Sensor Networks, in *Proceedings of the International Conference on Very Large DataBases (VLDB'04)*, 2004, pp. 588-599.
[21]. S. K. Singh, 2011, Database Systems: Concepts, Design and Applications, Second edition, *Pearson Education;* 2011.
[22]. C. S. R. Prabhu, Object – Oriented Database Systems: Approaches and Architectures, *Prentice-Hall of India Pvt. Ltd.*, 2010.
[23]. Sibsankar Haldar, SQLite Database System: Design and Implementation, *Motorola Mobility, Inc.*, 2011.
[24]. M. Xiong, J. A. Stankovic, K. Ramamritham, D. F. Towsley, R. M. Sivasankaran, Maintaining Temporal Consistency: Issues and Algorithms, in *Proceedings of the International Workshop on Real-Time Database Systems*, 1996, pp. 1–6.
[25]. Bulent Tavli, Wendi B. Heinzelman, Mobile Ad Hoc Networks - Energy-Efficient Real-Time Data Communications, *Springer*, 2006, pp. I-XXI, 1-265.
[26]. F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, Cross-Level Sensor Network Simulation with COOJA, in *Proceedings of the Annual IEEE Conference on Local Computer Networks*, Tampa, Florida, USA, 2006, pp. 641- 648.
[27]. Abderrahmen Belfkih, Bruno Sadeg, Claude Duvallet, Laurent Amanton, Study of Timing Properties on Data Collection and Query Processing Techniques in Wireless Sensor Networks, in *Proceedings of the International Conference on Sensor Networks (SENSORNETS'15)*, Angers, France, 2015, pp. 77–84.