Study of Timing Properties on Data Collection and Query Processing Techniques in Wireless Sensor Networks

Abderrahmen Belfkih, Bruno Sadeg, Claude Duvallet and Laurent Amanton
University of Le Havre, BP 1123, 76063 Le Havre cedex, France

Keywords: Wireless Sensor Network, Data Collection, Query Processing, Sensors Database, Time Constraints.

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 have 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.

1 INTRODUCTION

Wireless sensor networks 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 for the WSN reflects 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 much importance to the temporal constraints including the data freshness constraint and the deadline constraints.

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 (Bhattacharya, 2008). In real-time applications, data freshness is a key performance factor. It is important to properly 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 (Fung et al., 2002) and TinyDB (Madden et al., 2003) 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. (García-Hernando et al., 2008).

In this paper, we are interested in investigating two approaches for data processing in WSN (i) a query processing approach using an abstract database and (ii) a periodic data collection. We simulate temporal constraint 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: (i) 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, (ii) the data collection time. The fastest is this parameter, the more fresh are data, (iii) the database response time, which can improve the data arrival time to the user.

The remainder of this paper is structured as fol-

Belfkih A., Sadeg B., Duvallet C. and Amanton L..
Study of Timing Properties on Data Collection and Query Processing Techniques in Wireless Sensor Networks.
DOI: 10.5220/0005273200770084
In *Proceedings of the 4th International Conference on Sensor Networks* (SENSORNETS-2015), pages 77-84
ISBN: 978-989-758-086-4

lows. Section 2 reviews related work. In Section 3, we describe the two approaches cited previously. In Section 4, we carry out simulations, thereafter 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 WORK

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

2.1 Data Freshness in WSN

Suriyachai *et al.* have discuss the time-critical data delivery in WSN (Suriyachai et al., 2010). They have presented a novel TDMA-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) (Sharaf et al., 2003). 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, the following query 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.

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

Choi *et al.* consider that the aggregation time in WSN can be affected by the deadline constraints, i.e, data validity period or data delivery deadline (Choi et al.,). They have proposed a new aggregation time control (ATC) algorithm, which adjusts 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 (Hiromori et al., 2012). DD transforms the network to a set of trees, rooted at the sinks. Then, it adjusts them, in order 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 researches 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.* (Bonnet et al., 2001). They consider the WSN as a relational database that can be queried using *SQL-like* queries. Abstract databases present a declarative approach to facilitate the description and processing of queries in the WSNs. Data required by the base station form a virtual table, in which columns rerepresent the data requested by the user.

Madden *et al.* have presented a new query processing system, named TinyDB, using TinyOS (Levis et al., 2005) as an operating system, for extracting information from a network. It is composed of two subsystems: (i) TinyDB application, running on each node of the network, (ii) 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 network. The users writes the query that will be compiled and split into tasks, then injected into the network. Once a node receives the query, it processes it, 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:

```
SELECT AVG (temperature)
FROM sensors
WHERE light > 400
EPOCH DURATION 10s
```

TikiriDB (Laxaman et al., 2010) is a database abstraction layer for ContikiOS. It provides function-

ality to collect data from WSNs 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 injects them in WSN. It receives thereafter the results, which will be forwarded to the end user by the SF application. In the following, here is an example of TikiriDB query which permits to get the light and the temperature from all sensors whose temperature is above 50. Each new reading will be delivered to the end user every 2 seconds for a duration of 10 seconds.

```
SELECT light,temperature
FROM sensors
WHERE temperature > 50
SAMPLE PERIOD 2s FOR 10s
```

Amato *et al.* have proposed MaD-WiSe (MAnagement of Data in WIreless SEnsor networks) (Amato et al., 2010), 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.

Corona (Khoury et al., 2010) is a distributed in-network query processing system which provides a declarative query language SQL-Like 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 capable of running multiple applications simultaneously.

An example of Corona query is given in the following:

```
SELECT temperature
FROM sensors
WHERE light > 100
EPOCH 60s
FRESHNESS 10s
```

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

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 (Khoury et al., 2010) . 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: (i) sensor nodes which report informations about the environment, (ii) a base station which collects periodically data from sensor nodes, (iii) a remote database used to store the received data, (iv) 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 queries responses.

3.1 First Scenario: Data Collection with Remote Database

In this scenario, 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 send them to the 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.

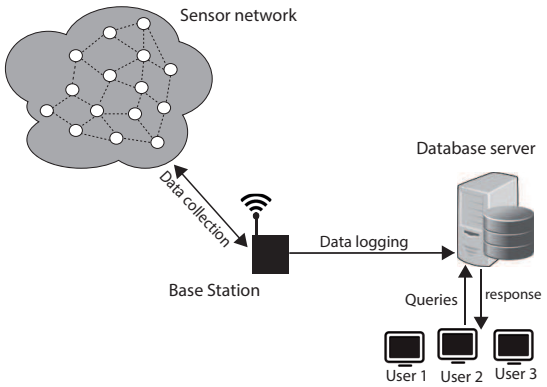

Figure 1: Data collection with remote database.

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. User specifies the data it want to collect from WSN, through SQL-like declarative queries. It sends

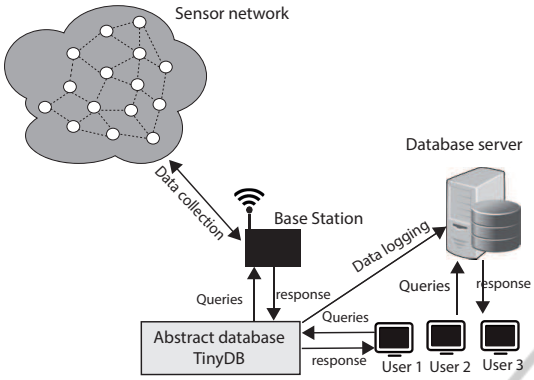

Figure 2: Query processing with abstract database.

them to the base station via the 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. These parameters are: (i) the network convergence time, an important factor to ensure the temporal constraints in WSN. It precedes the process of data collection. It can affect on the arrival time of data when it takes a long of time. We consider also (ii) the data collection time, which presents 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. Finlay, (iii) 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. We will show in the next section (simulations) that sometimes, data freshness cannot be met in this stage.

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 & RESULTS

4.1 Simulation Environment

4.1.1 First Scenario

We have used COOJA (Österlind et al., 2006), the network simulator for ContikiOS, 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 a area within 100m*100m and the sink is placed outside of the deployment area. The number of sensors vary 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)	100x100
Number of nodes	10-100
Number of sink Nodes	1
Source data rate (events/s)	Random
Radio range (m)	80
Data packet size (bytes)	15

We use RPL (Routing Protocol for Low-Power and Lossy Networks) (Winter et al., 2012) 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 and DSDV (Vucinic et al., 2013). It provides efficient routing paths guaranteeing data delivery before deadline in WSN. We have used ContikiMAC radio duty cycling protocol (Dunkels, 2011) inspired from existing duty cycling protocols. It has shown efficient time management because it uses a wake-up mechanism to defines a precise timing between packet transmission. We have used the default Contiki MAC layer CSMA (carrier sense multiple access) which is the only one MAC driver in Contiki. The table 2 below summarizes the different layers protocol 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 Second 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 managing. 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 are 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	Mica2
Simulation area(m)	100x100
Number of sender nodes	10-100
Number of sink Nodes	1
Radio range (m)	80
Data packet size (bytes)	20

TinyDB uses a tree-Based Routing Protocol (Gehrke and Madden, 2004) 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 will receive the query. TinyDB uses CSMA protocol, which is the already existing default MAC protocol used by TOSSIM. 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 below summarizes the different layers protocol 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 First Scenario

Each sender node uses the UDP sender algorithm to send its data to the base station. We have defined a transmission period of 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. We have activated the digital sht11 sensor, relative to sensor Humidity and temperature. A sender node listens to its neighbors using tcpip_handler() function and continues to send data currently detected to the base station using the "send_packet" function.

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 = tcpip_event then
6:          tcpip_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 Second Scenario

When TinyDB receives the query from the user, it parses and translates it to SQL query into an execution plan. Then, it injects it into the network via the base station. Each node receives the query, processes it using a query processor and returns the requested values. 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 arrival time of data. The shorter is the convergence time, the quicker is the availability of the data. In Figure 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) message. The nodes 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

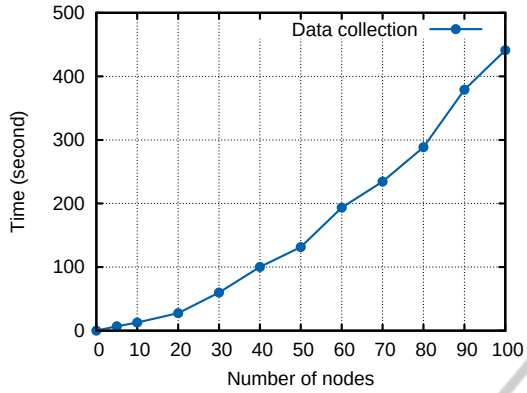

Figure 3: Network convergence time for data collection.

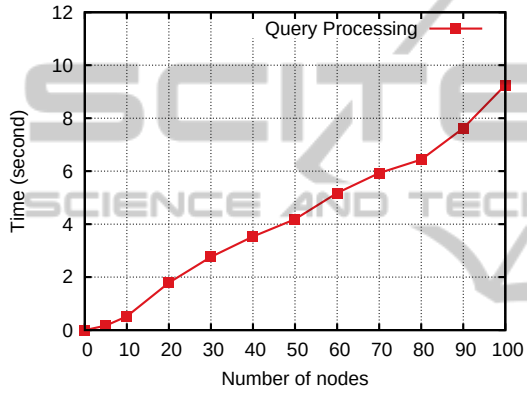

Figure 4: Network convergence time with TinyDB.

in Figure 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). 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 transmit, 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 affects 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 and the path quality depend 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.

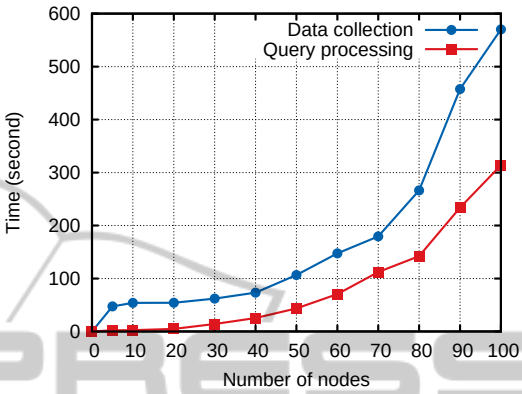

Figure 5: Completed Cycle Time.

Figure 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 to join or not the structure.

We also notice in Figure 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 for the data listened to them. The whole process is repeated for each period and query.

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 peri-

Table 5: Duplicate sends with periodic data collection and TinyDB query processing.

	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

odic data collection. The MAC-layer protocol can be considered as one of the causes to explain these results. Tossim implements CSMA[1] based MAC protocols 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 backoff, 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 backoff 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 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 precise timing between transmissions (Dunkels, 2011), which explains the decrease in duplicate Packets Sending.

4.3.4 Impact of Choosing the Database on Average Response Time

We use in this test three DBMS[2]: PostgreSQL, MySQL, SQLite, to evaluate queries response time. Table 6 shows that SQLite database gives the best average response time for the SELECT queries, then we find 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 reopen the database file, and invalidate its cache, for each transaction. It does not allow multiple concurrent writes in the same time. PostgreSQL allows multiple transactions to proceed with inserts concurrently. MySQL is better than PostgreSQL for reading tables because it has a ”Query Cache” to make queries faster. SQLite is faster than MySQL because it uses a disk access to read and write directly from the database files on disk.

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 Network Topologies on Data Collection Time

We have tested 15 sender nodes and one sink node with four network topologies. The result in Table 7 shows the time spent to collect data from all sensors for the two technologies.

Table 7: Complete cycle time (ms).

Topologies	Data collection	Query processing
Star	62312	1832
Mesh	56002	1362
Grid	54121	2163
Tree	53515	2143

We note that the tree network topology for data collection technology is more 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 to define DODAG networks and to build path to the root. It can not fully exploit, which affects the quality of paths. 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 an intelligent routing decisions, where every node selects a parent based on link quality to the base station.

5 CONCLUSION

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

[1]Carrier sense multiple access

[2]DataBase Management System

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

Amato, G., Chessa, S., and Vairo, C. (2010). Mad-wise: A distributed stream management system for wireless sensor networks. *Softw. Pract. Exper.*, 40(5):431–451.

Bhattacharya, S. (2008). *Achieving Application Quality of Service in Resource-constrained Wireless Sensor Networks*. PhD thesis, St. Louis, MO, USA.

Bonnet, P., Gehrke, J., and Seshadri, P. (2001). Towards sensor database systems. In *Proceedings of the International Conference on Mobile Data Management*, pages 3–14, London, UK, UK. Springer-Verlag.

Choi, J. Y., Lee, J., Lee, K., Choi, S., Kwon, W. H., and Park, H. S. Aggregation time control algorithm for time constrained data delivery in wireless sensor networks. In *Proceedings of the 63rd Vehicular Technology Conference*.

Dunkels, A. (2011). The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science.

Fung, W. F., Sun, D., and Gehrke, J. (2002). Cougar the network is the database. In *International conference on Management of data*, pages 621–621, New York, NY, USA. ACM.

García-Hernando, A., Martínez-Ortega, J., López-Navarro, J., Prayati, A., and Redondo-López, L. (2008). *Problem Solving for Wireless Sensor Networks*. Computer Communications and Networks. Springer.

Gehrke, J. and Madden, S. (2004). Query processing in sensor networks. *IEEE Pervasive Computing*, pages 46–55.

Hiromori, A., Uchiyama, A., Yamaguchi, H., and Higashino, T. (2012). Deadline-aware data collection in csma/ca-based multi-sink wireless sensor networks. In *International Conference on Mobile Computing and Ubiquitous Networking*, pages 1–7.

Khoury, R., Dawborn, T., Gafurov, B., Pink, G., Tse, E., Tse, Q., Almi'Ani, K., Gaber, M. M., Röhm, U., and Scholz, B. (2010). Corona: Energy-efficient multi-query processing in wireless sensor networks. In *DASFAA (2)*, pages 416–419.

Laxaman, N., Goonatillake, M., and De Zoysa, K. (2010). Tikiridb: Shared wireless sensor network database for multi-user data access. *CSSL*.

Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., and Culler, D. (2005). TinyOS: An Open Operating System for Wireless Sensor Networks. In *Ambient Intelligence*, volume 35, pages 115–148. Springer Berlin Heidelberg.

Madden, S., J. Franklin, M., M. Hellerstein, J., and Hong, W. (2003). The design of an acquisitional query processor for sensor networks. In *Proceedings of the international conference on Management of data*.

Österlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. (2006). Cross-level sensor network simulation with COOJA. In *The 31st Annual IEEE Conference on Local Computer Networks, Tampa, Florida, USA, 14-16 November 2006*, pages 641–648.

Sharaf, M. A., Beaver, J., Labrinidis, A., and Chrysanthis, P. K. (2003). Tina: A scheme for temporal coherency-aware in-network aggregation. In *Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access*, MobiDe '03, pages 69–76, New York, NY, USA. ACM.

Suriyachai, P., Brown, J., and Roedig, U. (2010). Time-critical data delivery in wireless sensor networks. In *Distributed Computing in Sensor Systems*, pages 216–229. Springer Berlin Heidelberg.

Vucinic, M., Tourancheau, B., and Duda, A. (2013). Performance comparison of the rpl and loadng routing protocols in a home automation scenario. In *WCNC*, pages 1974–1979.

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., and Alexander, R. (2012). RPL: IPV6 Routing Protocol for Low-Power and Lossy Networks.