Accepted Manuscript

EATDDS: Energy-aware middleware for wireless sensor and actuator networks

Anas Al-Roubaiey, Tarek Sheltami, Ashraf Mahmoud, Ansar Yasar

PII: S0167-739X(18)33177-7
DOI: https://doi.org/10.1016/j.future.2019.01.060
Reference: FUTURE 4753

To appear in: *Future Generation Computer Systems*

Received date : 22 December 2018
Revised date : 23 January 2019
Accepted date : 27 January 2019

Please cite this article as: A. Al-Roubaiey, T. Sheltami, A. Mahmoud et al., EATDDS: Energy-aware middleware for wireless sensor and actuator networks, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2019.01.060

Available online at www.sciencedirect.com

ScienceDirect

Journal homepage: www.elsevier.com/locate/aebj

EATDDS: Energy-Aware Middleware for Wireless Sensor and Actuator Networks

Anas Al-Roubaiey[a], *Tarek Sheltami*[a, *], *Ashraf Mahmoud*[a], *Ansar Yasar*[b]

[a]*Computer Engineering Department, King Fahd University of Petroleum & Menirals, Dhahran, 31261, KSA*
[b]*Transportation Research Institute, Hasselt University, Diepenbeek,B-3590, Belgium*

ARTICLE INFO

Article history:
Received 00 December 00
Received in revised form 00 January 00
Accepted 00 February 00

Keywords:
Middleware
Internet of things
WSN, Sensor Actuator Network
Data Distribution Service
Energy-Aware

ABSTRACT

According to Object Management Group organization, Data Distribution Service (DDS) middleware is the leading technology for Industrial Internet of Things (IIoT). Therefore, using DDS-based middleware for Wireless Sensor/Actuator Networks (WSAN) will extremely ease the development and integration of WSAN applications into IIoT, which has an effective impact on improving the productivity and saving the cost. However, applying such technology over WSAN significantly affects the energy consumption. In this work, an energy-aware middleware for WSAN is developed based on DDS standard, which is called EATDDS. Furthermore, developing this middleware leads to a major enhancement into TOSSIM simulator; in which an Online Energy Model (OEM) is developed to make TOSSIM capable of developing and testing energy-aware protocols. The model is validated by comparing it against POWERTOSSIM-Z. Results show that EATDDS is efficient and can be accommodated with limited system resources

1. Introduction

Adding actuators to the traditional Wireless Sensor Networks (WSNs), made them more effective and smart. Figure 1 shows the basic three-layer architecture for Wireless Sensor/Actuator Networks (WSANs), where the data is sensed in the sensor/actuator layer, gathered in the Sink (Base Station) layer, and finally manipulated in the application layer. Nowadays, (WSANs) play the basic role in developing monitoring and control systems, for example, Internet of Things, Industrial Internet of Things, industrial process automation, smart cities, healthcare systems, and structural health monitoring [31]. Due to the large number of different platforms in such applications, middleware technology becomes very important to smoothly exchange the data between these systems as well as simplifying the application development over such heterogeneous platforms. While the energy is a very critical resource in WSAN, adding a middleware technology is a very challenging issue.

In this work, Data Distribution Service (DDS) [1] [2], a promised middleware technology standardized by Object Management Group (OMG), is selected to be the base of our proposed energy-aware protocol. DDS is the standard of the publish/subscribe communication model [3], where the DDS is mainly a data centric middleware (data centricity is extremely well-suited WSAN applications). Currently, DDS is widely being used in both academic and industry fields. Besides its benefits as a middleware, it supports many Quality of Services (QoS) polices, which makes it the leading technology in Industrial Internet of Things, based on OMG organization. TinyDDS [4] is a lightweight version of DDS aimed at adapting the DDS middleware to work over WSAN. However, this

* *Corresponding author*. Tel.: +966-13-8604678; fax: +966-13-8603950.
E-mail address: tarek@kfupm.edu.sa

Peer review under responsibility of xxxxx.

Hosting by Elsevier

xxxx-xxxx/$ – see front matter © 2014 xxxxxxxx. Hosting by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.aebj.2014.10.012

work ported just the main functionality of DDS without any QoS support and energy-aware mechanisms.

The contribution of this paper is threefold: Frist, before adding any modification into TinyDDS middleware, an extensive performance evaluation for TinyDDS over telosB platform [5] is conducted. As a result, the exact resources consumption, specifically in terms of memory and energy, is determined. The second contribution is making TOSSIM simulator [6] [7], a well-known TinyOS [8]SIMulator for wireless sensor networks, capable of simulating any energy-aware mechanism. This contribution is achieved by developing an Online Energy Model (OEM) that enables TOSSIM to take energy measurements while the simulation is running. The main contribution is the Energy-Aware TinyDDS (EATDDS) protocol, which extremely ensures an even energy consumption over the network nodes.

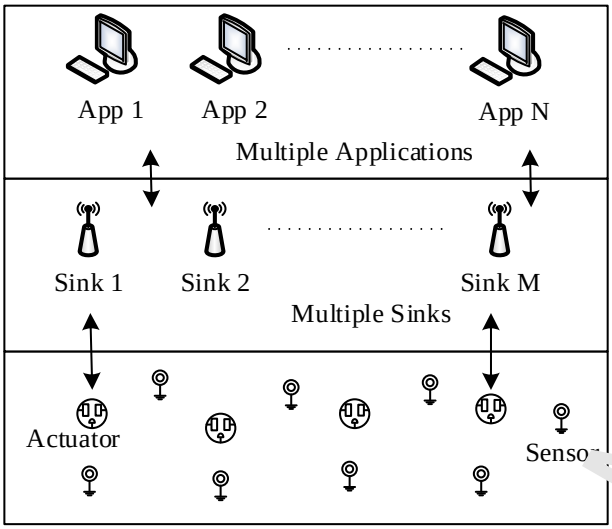

Fig. 1. Three layers' architecture of sensor/actuator networks

The rest of the paper is organized as follows. In section 2, we review the literature for middleware technology in WSAN. Section 3 introduces the Online Energy Model of TOSSIM simulator. The EATDDS protocol is presented in section 4 and evaluated in section 5. Finally, the paper is concluded in section 6.

2. LITERATURE REVIEW

Since DDS is basically a pub/sub communication model, in this section we discuss the pub/sub based solutions that are specifically proposed for WSAN recently.

Directed Diffusion [9] is considered the earliest pub/sub communication paradigm for wireless sensor networking. It is a data-centric protocol in that all the communication is for named data that is described by attribute-value pairs. As any pub/sub system, it has almost the same common elements and functions. The subscriptions are called interests and are broadcasted throughout the whole network. During the subscriptions dissemination, gradients are set up in every pair of neighboring nodes. Gradients specify a data rate and a direction in which to send a publication/event. These gradients are used later to draw the events. Each node examines the interest and does a matching process locally. If it has the requested data, then it sends back the information to the sink by reinforcing the reverse of the path of the interest. Otherwise, it propagates the interest through the network. Thus, the matching process is distributed and avoids the centralized processing approach, which is not suitable for sensor networks. This is because it does not evenly distribute the energy consumption. However, this approach adds an overhead in terms of memory, processing and communications, since all the nodes have to do the same process for each interest. Intermediate nodes can cache interests and use them to be directly forwarded based on previously cached data. Also, they do in-network data aggregation to minimize the data traffic and thus consume less energy. The data is represented using structures in the form of attribute-value pairs, and these attributes can be filtered to get specific information (content filtering). For each received interest, there is a gradient associated with it, which is a direction state that is directed towards the node from where the interest is received with specified data rate. Recently, a secured version of this protocol has been proposed, which provides authentication and integrity checks with relatively low overhead [10].

MQTT-S [11] is an IBM pub/sub protocol that was invented by Stanford-Clark and Hunkeler in 1999, and named as Message Queuing Telemetry Transport (MQTT) [12]. It is a simple and lightweight messaging protocol designed for constrained devices with ***low***-bandwidth, and for high-latency or unreliable networks. Consequently and due to its mentioned lightweight properties, they proposed an extension version of MQTT protocol to be suitable for wireless sensor networks [13]. The main goal was to simplify the integration of the WASN with the enterprise networks by extending the enterprise pub/sub middleware protocol into the WSN infrastructure. Unlike Mires, the pub/sub service, referred to as the notification service in MQTT, is located in brokers that uses the original MQTT protocol, where the Sensor/Actuator (SA) devices software is kept as simple as possible. The SA devices use the collection tree protocol (CTP) [14] as its underlying routing protocol, which allow any device to send data to the closest gateway. Three levels of reliability are provided by this implementation, including: (1) best effort, where the publisher sends just once either successfully received or not, (2) retransmit until the message is acknowledged (may incurs redundancy), and (3) assure no redundancy. One drawback of this approach is ***that*** the broker architecture raises the centralized approaches problems such as being a single point of failure and a bottleneck. Also, the translation from gateways (MQTT-S) to broker (MQTT) causes more delay, which increases the potential of considering this protocol being unsuitable for real-time systems.

TinyCOPS [15] is a component-based middleware that provides a well-defined content-based pub/sub service to WSN. It simplifies the selection and composition of the components, which allows the application designer to easily adapt the service by making orthogonal choices about the following: (1) communication protocol components for subscription and notification delivery, (2) supported data attributes, and (3) set of Service Extension Components (SEC). SECs are decoupled from the TinyCOPS core in which it can be reusable in different applications and platforms. Two different types are supported: Communication SEC (CSEC), which adds services to the communication protocol and Attribute SEC (ASEC), which adds services to the endpoints. Similar to the directed diffusion, it uses an attribute-based naming scheme. This scheme is augmented with metadata information provided through the pub/sub Application Programming Interfaces (API) and used to send control information to the publisher, e.g. sensing rate, and to add additional communication control information, e.g. timestamps, message sequence number, etc.

TinyDDS [4] is the adopted version of OMG DDS standard for WSN. It is a lightweight pub/sub middleware that allows applications to interoperate across the boundary of WSNs and access networks, regardless of their programming languages and protocols. Moreover, in the application and middleware level, it allows WSN applications to be flexibly customized to meet the application requirements, and to have fine-grained control on it. It can adaptively perform event publication according to dynamic network conditions and autonomously balances ***its*** performance among conflicting objectives using an evolutionary multi-objective optimization mechanism. The main contributions of TinyDDS to WSNs are (1) providing interoperability with access networks, and (2) adding flexibility to customize non-functional properties such as data aggregation, event filtering, and routing. Although TinyDDS provides efficient services and support for WSN, a complete and robust DDS-based system for WSAN is yet to be developed [16]. TinyDDS lacks energy saving mechanisms and energy consumption evaluation, because it is still not lightweight enough to fit the WSAN requirements.

UPSWSN-MM stands for Ubiquitous Publish/Subscribe platform for WSN with Mobile Mules. It is an application-specific pub/sub middleware with content-based subscription model [17]. Its architecture comprises stationary part represented by WSN, and mobile part represented by mobile phones. Ubiquitously, Internet users can ***access*** the WSN data anytime from anywhere through a broker-based platform, i.e. a server. The sensors are distributed over the monitored area and publish data to the mobile phones, which then send it to the interested subscribers or Internet clients via mobile phone networks, e.g. 3G. The proposed solution was tested via outdoor test-bed, and a hiking trial monitoring application is developed on top of the PSWSN-MM middleware. In its reliability implementation, the packet does not be sent until the previous one is acknowledged. Because of that, the system is not suitable for real-time systems. Moreover, the system lacks other QoS mechanisms support such as priority and deadline.

PS-QUASAR [18] is a pub/sub middleware that focuses on providing high level programming model and QoS support; specifically, reliability, deadline, and priority to the WSAN applications. In this solution, all nodes in the network are potential publishers of each of the topics. PS-QUASAR also handles a many-to-many exchange of messages between nodes in a fully distributed way by means of multicasting techniques. It is composed of three different modules: maintenance protocol, routing module, and APIs. The maintenance protocol is in charge of creating the links between neighbor nodes, and discovering pub/sub end nodes, i.e. publishers and subscribers. The routing module to route the events uses the information collected from the maintenance protocol. A topic-based programming model is used to provide a set of APIs for developers to develop WSAN applications using PS-QUASAR middleware. The Bellman-Ford algorithm [19] is enhanced and used to build a routing tree where each node maintains a routing table (memory overhead). Although PS-QUASAR provides QoS-aware, energy efficient, and robust protocol, the cost in such mechanisms would be in the expense of memory space. One of the most recent work in this area is in [20], where the authors claim that the proposed publish/subscribe middleware is an energy efficient. However, the paper doesn't include any details about the energy consumption mechanisms and the proposed system architecture is almost the same as the DDS standard architecture.

SSPSM is a channel-based scalable, Self-Stabilizing Publish/Subscribe Middleware that particularly meets the requirements of WSNs [30]. In this work, the authors consider message and memory corruptions while respecting dynamic network changes, such as, node and link removals and additions. The middleware aims to capture the trade-off between the scalability of the overlay network (i.e., the size of the routing tables) and the message routing overhead incurred by nodes forwarding publications. As optimal solutions (e.g., Steiner tree based overlays) are too costly to build and maintain, they use a simpler structure of a virtual ring. To reduce message complexity channel specific shortcuts are used on this ring. With the help of a neighborhood management protocol the system suffices with an average storage demand per node that is only proportional to the number of channels, i.e., independent of the network size.

PSMSWSN is a Publish/Subscribe Messaging System for Wireless Sensor Networks [29]. It is an implementation of topic-based publish/subscribe model in wireless sensor networks. There are two main phases in this model; the initial phase contains the steps of distribution of subscription messages and the second phase contains the steps of distribution of publish messages.

Table 1. Pub/sub middleware solutions in WSAN

Solution	Sub Scheme	Overlay Infra-structure	Multiple Sinks	Actuator Support	QoS			Energy Awareness	Mobility
					R	P	D		
Directed Diffusion (2003)	Topic/ content based	P2P	Y	N	N	N	N	Y	Y
TinyCOPS (2008)	Content based	Broker	Y	N	N	N	N	Y	Y
MQTT-S (2008)	Topic based	Broker	N	Y	Y	N	N	N	N
TinyDDS (2009)	Topic/ Content based	Broker	Y	N	Y	Y	Y	N	N
UPSWSN-MM (2012)	Content based	Broker	Y	N	Y	N	N	N	Y
PS-QUASAR (2013)	Topic based	P2P	Y	Y	Y	Y	Y	Y	N
SSPSM (2015)	Topic based	P2P	Y	Y	N	N	N	N	N
PSMSWSN (2016)	Topic based	P2P	Y	N	N	N	N	N	N
EATDDS	Topic/Content based	Broker	Y	Y	Y	Y	Y	Y	N

The above solutions are compared regarding their pub/sub properties, provided services, and supported QoSs. Table 1 shows the detailed comparison. The exact criteria that are used in comparing these solutions are: 1) subscription scheme: where two main schemes are used, topic and content based. The content based scheme gives more control to the subscribers on the required data. 2) Overlay infrastructure: either centralized, i.e. broker-based, or decentralized, i.e. P2P. 3) Multi-sinks and actuator support. 4) QoSs: our focus on the reliability, priority, and deadline that is based on the reviewed solutions. 5) Energy aware: whether it includes energy conservation mechanisms. 6) Mobility support.

3. ONLINE ENERGY MODEL

TinyDDS is implemented over TinyOS code, therefore, the main challenging issue in this work is how to use TOSSIM to develop our energy aware protocol EATDDS. In this section, we elaborate on our proposed Online Energy Model (OEM), and shed light on its implementation and validation. Online means while the simulation is running it has the capability to measure the energy consumption of the network.

TinyDDS is implemented over TinyOS code, therefore, the main challenging issue in this work is how to use TOSSIM to develop our energy aware protocol EATDDS. In this section, we elaborate on our proposed Online Energy Model (OEM), and shed light on its implementation and validation. Online means while the simulation is running it has the capability to measure the energy consumption of the network.

One of the most well-known and accurate simulators for wireless sensor networks is TOSSIM [6] [7], an event-driven simulator for TinyOS applications. However, no energy measurements are supported by TOSSIM, which is considered as a major shortage in a WSN simulator since the energy consumption is a very important metric in the performance evaluation of any WSN protocol or application. Therefore, two main extensions are developed to tackle this problem by integrating energy measurements' tools into TOSSIM. These extensions are: POWERTOSSIM [21] and POWERTOSSIMZ [22], however; POWERTOSSIM is for mica2 platform and TinyOS 1x, whereas POWETOSSIMZ ports the model to TinyOS 2x, and micaZ platform. Both simulators work by accurately tracking the power states of each component in TOSSIM simulator, e.g. Microcontroller unit (MCU), Memory, LEDS, and Radio, during the whole period of simulation run. At the end of the simulation, the output file from these energy simulators is subjected to post processing to compute the final results of simulation energy consumption of each component. The post processing process of the POWERTOSSIM and POWERTOSSIMZ depends on the energy measurements from the mica2, and micaZ datasheets [23] respectively.

The main limitations of the POWERTOSSIM and POWERTOSSIMZ are: (1) they support the mica series platforms only, (2) they do not support online energy measurements since they compute the final energy measurements after the simulation run. The latter issue is very important for any energy aware simulation study, because in such protocols the energy level of the network nodes should be known instantaneously to take the proper action according to the energy readings. Therefore, one of the challenging issues in this work is to come up with an energy model that allow us to develop and test our proposed energy-aware protocol EATDDS. In this chapter, we describe in details the proposed energy model that is used in our simulations and its implementation in TOSSIM components. Furthermore, we validate our model by comparing our results with the previous work PWOERTOSSIMZ.

3.1. OEM Description

Unlike POWERTOSSIMZ, in Online Energy Model (OEM) we focus on the Radio and MCU components only, since they are the most components that contribute in energy consumption, more specifically the Radio component. TinyOS is a component-based operating system, which consists of many components and these components are wired using interfaces that are either provided or used by a component. The TOSSIM simulator is part of TinyOS code; it consists of many components, where each mote unit, e.g. MCU and Radio, corresponds to one or more components. The main components that we use in online energy model implementation are the TossimPacketModel component which is corresponding to the Radio unit, and SimSchedularBasic which corresponds to the MCU unit. Figure 2 depicts the architecture of this model, as shown in the figure the power state tracking code is embedded into TOSSIM, and the energy model of the mote platform can be easily integrated into the simulator before a simulation run.

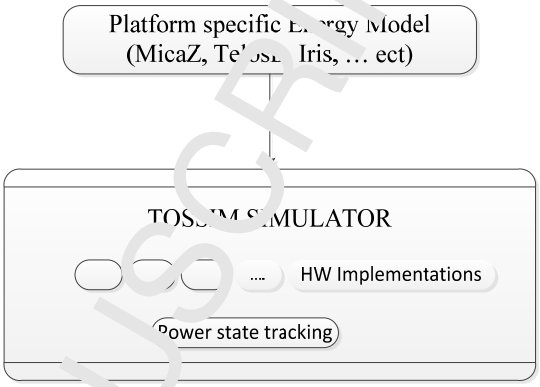

Figure 2. Online Energy Model Architecture

3.1.1. Radio Component

The radio is the largest energy consumer among all the other components in the mote. Both micaZ and telosB platforms use CC2420 Radio Chip. The corresponding component of the Radio in TOSSIM provides three main interfaces: Send, Receive and Split-control. In OEM we use Send and Receive interfaces to track the radio power states in TOSSIM simulator; specifically, in the TossimPacketModel.nc component. Three main states are tracked in the Radio component: Send, Receive and sleep. Thereby, the total energy consumption is calculated using equation 1, where the Δt represents the state duration (receiving, sending or sleeping), and V represents the used voltage, which is approximately 3 V, and I_{pstate} represents the consumed current of the power state, which is obtained from the energy model/datasheet of the used platform, e.g. as shown in Table 2. The OEM Radio algorithm is shown in algorithm 1.

$$EC_{Radio} = \Delta t * V * I_{Pstate} \qquad (1)$$

Table 2. Radio current consumption of micaz and telosb

MicaZ		TelosB	
Mode	**Current**	**Mode**	**Current**
Receive	19.7 mA	Receive	23 mA
Tx, -0 dBm	17.4 mA	Tx, -0 dBm	17.4 mA
Idle	20 uA	Idle	21 uA
Sleep	1 uA	Sleep	1 uA

Algorithm 1: Online Energy Consumption of the Radio

```
1: while sim ranning do {
2: while RoundTimer not fired do {
3:         if (send.start || receive.start)
4:                 eventstamp = simtime;
5:         if (send.done || receive.done) {
6:                 duration = simtime-eventstamp;
```

```
7:                  ActiveTime += duraiton;
8:                  if(send.done) ECs += duration * V *
Itx;
9:                  else
10:                 ECr += duration * V * Irx;
11:         }
12: }
13: IdleTime= SimTime - ActiveTime;
14: ECi += IdleTime * V * Iidle;
15: send (ECs,ECr,ECi);
16: reset RoundTimer;
17:}
```

Nomenclature
ECs: Sending Energy Consumption
ECr: Receiving Energy Consumption
ECi: Idle or Sleep Energy Consumption
V: source voltage;
Itx: transmission current
Irx: receiving current
Iidle: Idle current.
I: current

As mentioned above, this modification is on the core code of TOSSIM simulator. Now, to use the energy measurements online by TinyOS applications/protocols, we added a new component to represent the global energy measurements variables; this is due to the lack of supporting global variables in nesC [24]. As a result, these variables can be easily accessed by any component in the simulator during the simulation run.

3.1.2. Microcontroller (MCU) Component

To compute the energy consumption by MCU, it is important to track the amount of time the MCU spends in each MCU power state. Similar to the Radio component, equation 2 can be used to compute the energy consumption of MCU for each state. The current consumption of the MCU for MicaZ and TelosB motes are shown in **Error! Reference source not found.**. The main states that we use in our tests for MCU were Active and Idle states as illustrated in algorithm 2. As described earlier in this chapter, the MCU power state tracking code is integrated with SimSchedularBasic.nc component, specifically in the scheduler.runNextTask event. The main condition used in the MCU algorithm 2, if the scheduler has no tasks, then the MCU is in the idle state; otherwise it is in the Active state. The OEM of MCU algorithm is shown in algorithm 2.

$$EC_{MCU} = \Delta t * V * I_{Pstate} \qquad (2)$$

Algorithm 2: Online Energy Consumption of the MCU

```
1: while sim running do {
2: while RoudTime not fired do {
3:      duration = SimTime - PrevStateTime;
4:      if (PrevState == Active)
5:              ECactiveMCU= duration * V * IactiveMCU;
6:      if(PrevStateMCU == Idle)
7:              ECidle = duration * V * IidleMCU;
8:      if (nextTask = = noTask)
9:              PrevState = Idle;
10:             PrevStateTime = SimTime;
11:     } else {
12:             PrevState = Active;
13:             PrevStateTime = SimTime;
14:         }
15: }
16: Send (ECactiveMCU, ECidleMCU);
17: }
```

Table 3. MCU current consumption of Micaz and Telosb

MicaZ (ATmega128)		Telosb (MSP430)	
Mode	**Current**	**Mode**	**Current**
Active	8 mA	Active	1.8 mA
Idle	4 mA	Idle	54.5 uA
Sleep	9 uA	Sleep	5.1 uA

3.1.3. OEM Validation

Since the last extension for TOSSIM that enable energy measurements is the POWERTOSSIM-z (PTZ), we validate OEM with PTZ. Also, since PTZ cannot provide online results, we run the simulation of PTZ several times in order to get several points that we can use to compare against OEM.

The simulation scenario uses the default TinyDDS with Best Effort service, it includes five publishers and one subscriber, with transmission rate of one message per second; the simulation lasts for 120 minutes. The OEM measurements were taken during the whole simulation, whereas the PTZ measurements were taken at the end of several simulations with different times, i.e. 5, 35, 75, 120 minutes. Since we are testing the internal mote components, namely the Radio and MCU, we select one publisher node and take our energy measurements for both OEM and PTZ. The results are shown in Figure 3 and

Figure **4** for Radio and MCU respectively.

The energy model that we use in this validation is the MicaZ model. The radio component has just two power states, Transmission and receiving state. If it is not transmitting it switches to the receiving state, this is the default states in PTZ. On the other hand, the MCU has also two power states, active and idle.

The results show that the energy consumed according to the OEM for both Radio and MCU approximately increases linearly with time. That is because the data rate is constant and the network is very light, which means almost no probabilistic behavior that can change the consumption rate. This is adequate for our test since we are comparing two energy measurement tools, where any randomization can affect the comparison fairness. Table 4 shows the exact values of the results, only for the comparison points, i.e. 5, 35, 75 and 120 minutes. The error of the OEM relative to the results of PTZ is in the order of nano-Joule, which can be considered negligible.

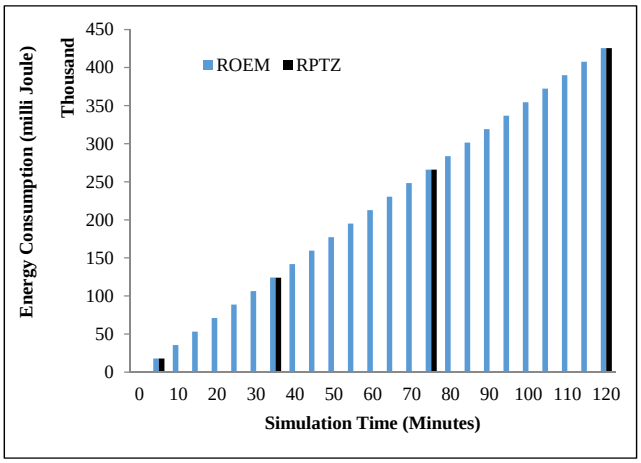

Figure 3. Energy consumption of the Radio Component

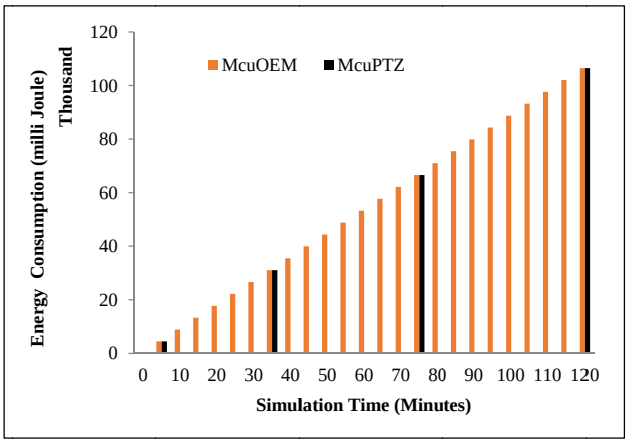

Figure 4. Energy Consumption of the MCU component

Table 4. The OEM and PTZ validation comparison -Radio

Sim. Time (sec)	Radio (mJ)		
	PTZ	OEM	Error %
5	17776.1	17727.92	-0.00271
35	124084.6	124095.4	8.71E-05
75	265907.2	265918.7	4.33E-05
120	425459.3	425470	2.51E-05

Table 5. The OEM and PTZ validation comparison -MCU

Sim. Time (sec)	MCU(mJ)		
	PTZ	OEM	Error %
5	4449	4454.685	0.00128
35	31056.3	31076.92	0.000 66
75	66552.1	66574.06	0.000
120	106485.3	106508.5	0.00022

4. EATDDS Protocol

In EATDDS protocol, we assume that the node location is known for all network nodes, e.g. using Global Positioning System (GPS) devices, or localization protocols. We use a grid topology, which is the tested topology in the original TinyDDS test. Since the energy consumption is directly proportional to the square distance between the sender and receiver [7], EATDDS uses the location of the nodes to minimize the distances between the publishers and interested subscribers, thus minimizing the energy consumption. The OEM is used in this work to monitor the energy consumption of the network nodes. Each node monitors its energy level and based on the common round used by all the nodes, it sends its information periodically to the cluster Rendezvous Node (RN) node.

EATDDS algorithm is inspired from LEACH-C protocol [25], where the network is considered as a cluster-based network. As we have three topics, i.e. three RN nodes, therefore, each RN node can form a separate cluster with all the publishers and subscribers that are relevant to that RN node. Figure 5 shows how the network can be clustered into three main clusters, each cluster represents a distinct topic. As shown in the figure, the topology that is used in EATDDS is a grid topology. In this particular scenario, three topics, each one has different shape (triangle, circle, and rectangle); three subscribers, and five publishers. The star subscriber (node 45) subscribes to all topics while the other two subscribers subscribe to the same shape publishers. It should be noted here that we used the ID-Based routing protocol in EATDDS [32].

In EATDDS algorithm, the selection of the RN node is based on the remaining energy as well as the distance from the RN to subscriber and relevant publishers. Each RN is responsible for one cluster, which has the same topic of the RN node. The network life time is divided into rounds; in each round the RN node selects a new RN node in its cluster. The new RN node is selected from the cluster nodes as the one having the maximum remaining energy.

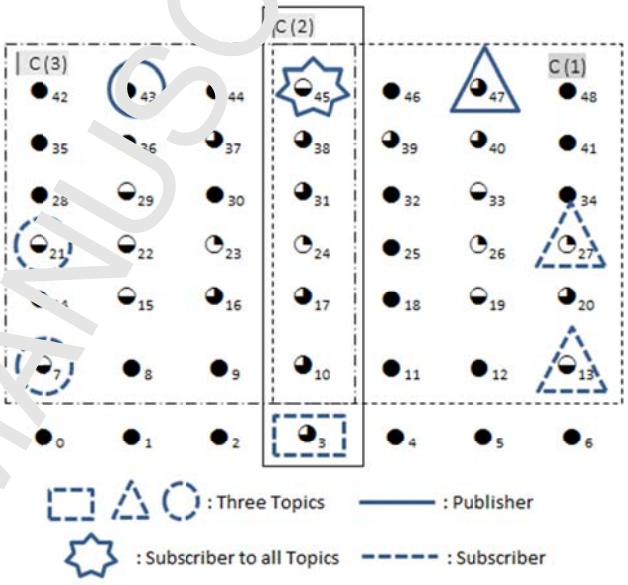

Figure 5. Cluster formation of EATDDS

Since all the nodes have registered the topic/data in the deployment phase, they can reach the main RN node, because, as discussed earlier, each topic is mapped to an RN node address. Thereby, it will be easy for those nodes to send their energy updates to the corresponding RN node periodically. In case there is more than one topic, which means more than RN nodes are existed, every node sends the energy updates to the all RN nodes in the network.

5. PERFORMANCE EVALUATION

This section introduces two main subsections: the first section answers a very important question relevant to the estimation of the overhead resulted from adding middleware technology into limited resources devices, i.e. sensors. The second section evaluates EATDDS middleware.

5.1. TinyDDS Overhead

In this experiment, we consider four main performance metrics: 1) Packet Delivery Ratio (PDR), which is defined as the number of packets successfully received by the subscribers over the number of packets sent by the publishers. As more overhead added to the network, the probability

of network congestion, buffer overflow, and thus packet dropping rate is expected to increase. 2) End-to-end delay metric, which is the average delay for all successfully received packets. Since this metric highly depends on the underlying protocols, we evaluate the two scenarios over the same underlying protocols to get more accurate results. 3) Energy consumption, where energy is a very important metric and critical issue in studying sensor networks. We compute the percentage of energy consumption by dividing the total consumption by the initial energy of the whole network. The initial energy of each node in the network was 2000 mAh, which is equivalent to 21600 Joules. 4) Memory space, where memory is a scarce resource in sensor devices that makes it also critical metric in evaluating sensor applications and protocols. Different platforms, namely: mica2, micaz, iris, and telosb, are evaluated in terms of memory space.

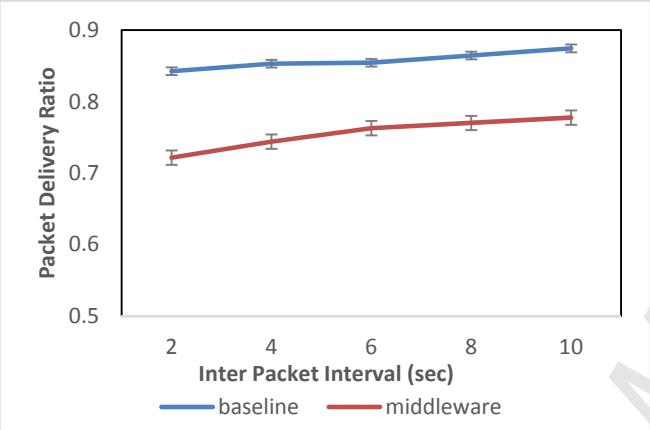

Figure 6. Packet delivery ratio comparison, baseline (without middleware) and middleware (TinyDDS)

Figure 6 shows the effect of the network load on the PDR on both tested scenarios. The PDR tests the network reliability. It should be noted that in this study our main concern is to evaluate TinyDDS overhead before EATDDS. Two applications are used, the baseline application, which is a simple data collection and dissemination application. The second application do the same function but with adding TinyDDS middleware. In the baseline scenario, the PDR varies very little with the Inter-Packet Interval (IPI), which means that the overall network load of the network is low. In contrast, the middleware scenario has larger variation than the baseline scenario with the increase of IPI, because it has more control traffic used in publisher, subscriber, and matching processes. The middleware overhead can be extracted from the drop of the network performance, represented by PDR value, which is nearly 10% compared to the baseline scenario. The error bars show the standard error for every single point in the results. This is added to show the level of accuracy of our results. For example, in middleware scenario, when IPI equals 6 the PDR mean value that is calculated from 10 runs is 0.76.

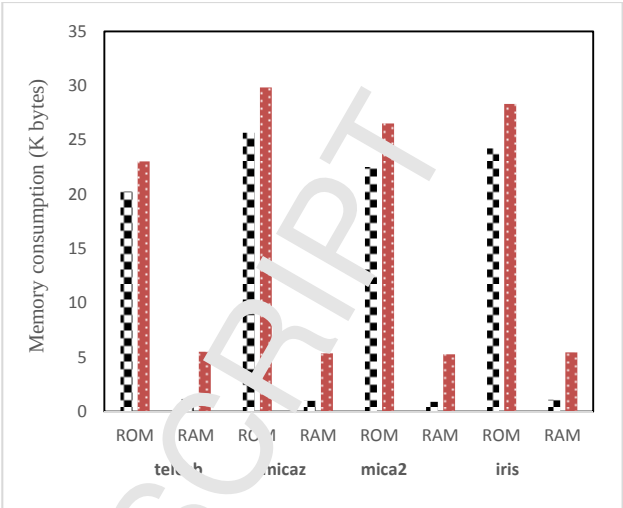

Figure 7. Memory cost comparison

From the PDR in Figure 6, we can see that the network in case of middleware scenario is more congested than in baseline scenario. As a result, the average packet end-to-end delay is higher in case of the middleware scenario as shown in Figure 8. The difference in the delay between both scenarios depends on the network traffic load, where the difference decreases as the IPI increases. That is because when the network is not overloaded, the packet delay almost the same when we have the same packet size. Thus, the figure shows that the difference in the delay nearly ranging from 60 ms (in case of 10 sec IPI) to 80 ms (in case of 2 IPI). Intuitively, the delay decreases as network load decreases (IPI increases). However, in case of the baseline scenario the end-to-end packet delay almost the same. That is because the network, in case of baseline scenario, has lightweight load and in all IPI values the packets reaches the base station using almost same number of hops. Whereas, in the other scenario, the network is overloaded, which results in more queuing delay and might be more hops due to network congestion.

The memory requirements in each scenario are illustrated in Figure 7. This figure describes the ROM and RAM consumption for four platforms: telosb, micaz, mica2, and iris. This figure includes the exact number of bytes needed by each scenario. For example, for the telosb platform, the baseline scenario uses 20270 bytes in program flash memory (ROM) and 1162 in RAM; whereas, the middleware scenario allocates 23034 bytes in ROM and 5512 bytes in RAM for the same telosb platform. Thereby, we can evaluate the memory overhead of a sensor device when a middleware is added. In telosb platform, the middleware overhead versus baseline application is about 14% more memory space in ROM and 3.7 times more memory space in RAM. This is considered a quite large memory space, relative to limited resources devices such sensor nodes. However, from telosb datasheet, these values are still acceptable where it has 48 KB ROM, and 10 KB RAM, and also 1 MB for logs, measurements readings, etc.

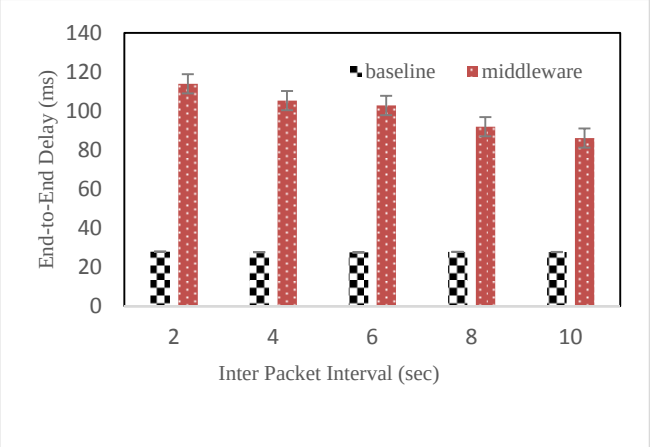

Figure 8. End-to-end delay comparison

The energy consumption evaluation is conducted using the POWERTOSSIMZ tool. POWERTOSSIMZ tool assumes each node has two AA batteries with capacity of 2000 mAh. In Figure 9, the energy consumption is computed as the percentage that has been consumed from fully charged batteries. For example, in case of 2 seconds IPI the total energy consumption of the network in the middleware scenario is 1.24% calculated from the total energy of the network; whereas, it is 0.87% in case of the baseline scenario. Due to the small interval of the simulation time, the total energy consumption is very small; however, clearly it shows the difference of energy consumption in both scenarios. In case of high traffic, 2 sec IPI, the middleware consumption is higher than the baseline scenario by 37%; whereas, in case of low traffic, 10 sec IPI, it is higher by 24% which means almost third of the network life time would be reduced when we use middleware technology in sensor networks.

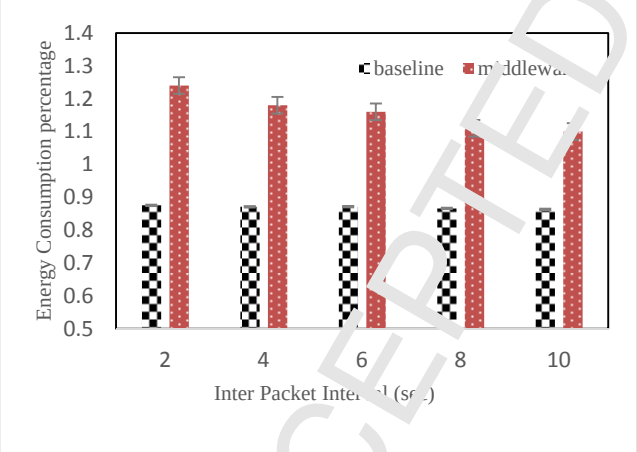

Figure 9. Energy consumption comparison

5.2. EATDDS Evaluation

In this section, EATDDS is extensively evaluated and tested under different network loads. The main focus in this evaluation is on the energy consumption metric and its related metrics, such as network life time and energy consumption per packet. Unlike the previous tests, this test is not limited by simulation time, in which we run the simulation until the first node dies, at this time the other measurements are taken.

5.2.1. Experiment setup

The simulation set up and network topology is the same as in the OEM, the topology can be shown in Figure 5. As mentioned above, the only difference is the unlimited simulation time, whereas in OEM simulations it was 1000 seconds, and in RTEDS it was 500 seconds, so it gradually increases. The new and most important parameter in this simulation is that we tracked the initial energy; where all the network nodes starts with an initial energy, and once this energy is dissipated the node is considered dead. We set the initial energy to be one joule, as in [25]. Moreover, the data rate is constant, that means all the protocols are subjected to the same workload, which makes a fair comparison. EATDDS round time is 350 second, which means every 350 second a new round is initiated by the main RN to change the distributed RN nodes.

5.2.2. Performance metrics

The focus in this evaluation is on the cost of the middleware in terms of energy consumption. In addition, the protocol performance is measured by how many successfully received packets per joule.

Network life time (NLT): the network life time is measured as the running time of the simulation until the first node dies. This occurs when the node consumes its whole energy, where the initial energy is one joule per node.

Packet per Joule (PPJ): this metric is a good indicator for the protocol efficiency in terms of energy savings. It is measured as the number of successfully received packets divided by the total energy consumption during the whole network life time.

Total Energy Consumption (TEC): the TEC is the summation of the energy consumption of all network nodes. All the energy measurements are in milli-Joule.

Wasted Energy (WE): it is the remaining energy after NLT occurs. A large amount of wasted energy reflects bad mechanism in terms of energy savings. It is measured by taking the summation of the remaining energy of the network nodes. Specifically, it is calculated by subtracting the total energy consumption from the total initial energy.

5.2.3. Results and analysis

The total energy consumption in Figure 10, and wasted energy in Figure 11 are the opposite of each other; the less energy consumption the more wasted energy. As shown in the two figures, the default TinyDDS appears to be worse than EATDDS, since it has the most wasted energy. Likewise, it has the largest total energy consumption that means less work has been done in this protocol. EATDDS protocol is better than the default TinyDDS when the work load is decreasing, that is obvious from the difference of the TEC that is increasing with IPI increases.

As mentioned earlier, the Packet per Joule measurement is a perfect metric for energy efficiency, the more packets per joule is the better. In Figure 12 , EATDDS protocol outperforms the default TinyDDS in case of less network load (Six IPI). Due to the random selection of the RN node, EATDDS may behave in a way that is inappropriate when subjected to heavy network load, this needs further investigation in future work.

In network life time, EATDDS shows a significant improvement against the default TinyDDS, as shown in Figure 13. Extending the network life time is the ultimate target from the any energy-aware protocol. In this results it reflects how EATDDS is good in distributing the energy consumption over the network load.

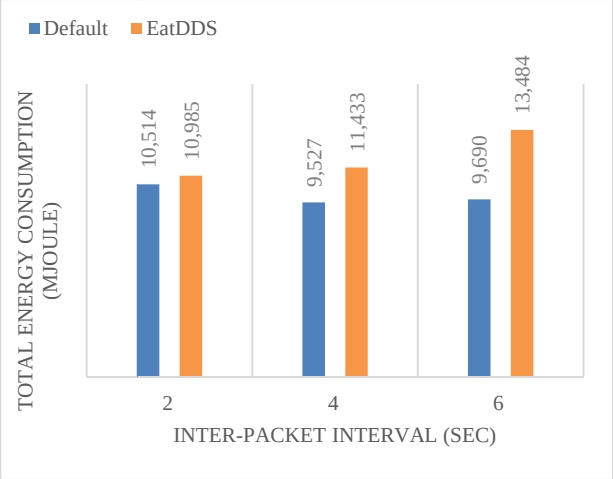

Figure 10. The network Total Energy Consumption

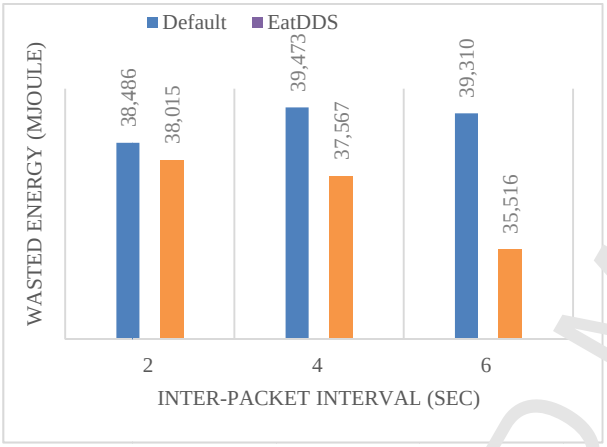

Figure 11. Remaining energy at the end of network lifetime

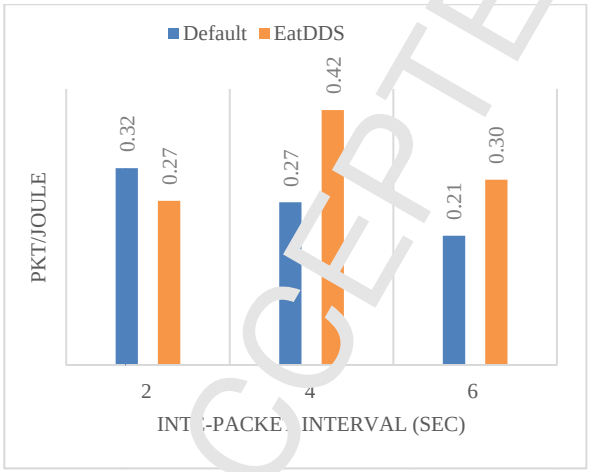

Figure 12. Packets per Joule vs. Inter-Packet Interval

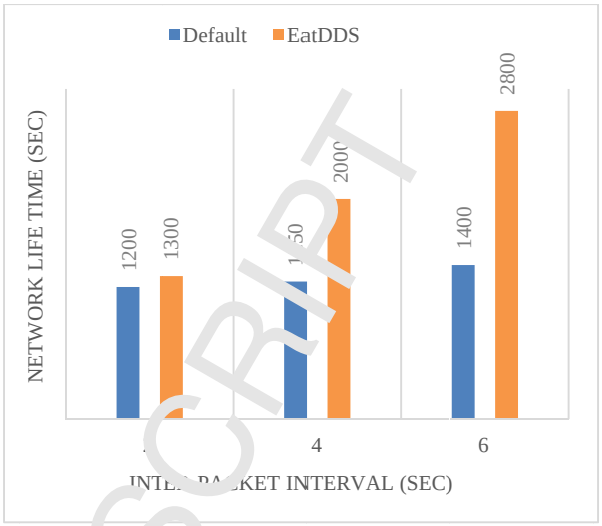

Figure 13. Network lifetime at the moment the first node dies

5.3. Prototyping

On one hand, the main advantage of TOSSIM simulator is that one can building a real implementation that needs minor modifications to be uploaded to real sensors and work normally. On the other hand, doing any modification in TOSSIM is very complicated. In this part, we introduce our prototype and how we tested the final version of EATDDS.

TelosB motes are used in this experiments, it is depicted in Figure 14. In this experiment we test the real energy consumption of TelosB platform utilizing EATDDS with different scenarios. In this test we use two scenarios: one centralized with RN and in the second scenario we remove the RN node to make the network fully distributed. The TelosB motes are used without the low power listing protocol, which means they are all the time in receive mode unless there is a transmission. Energizer batteries are used, and new ones are changed in every experiment.

Figure 14 TelosB mote platform

The seven nodes are distributed indoor, i.e. inside the lab as depicted in Figure 15. In each side, two publishers and one RN node; and the base station is placed directly on the USB port, as shown in Figure 16; however, also the base station were tested with new batteries to see the energy consumption in the base station nodes.

Figure 15 Experiment environment and testbed

Figure 16 The Base Station attached to the PC USB port

We evaluate the voltage versus the time, which represents the network life time, the memory and the end to end delay, i.e. from the publisher until it reaches the base station including passing the RN node. Table 6 shows the effect of the centralized approaches in real scenarios where the distributed scenario relaxes the network and thus minimizes the contention and consequently packet dropping and collisions. Furthermore, the standard deviation may reflect the instability of the centralized approach, since all publishers of the network have to go through this central RN.

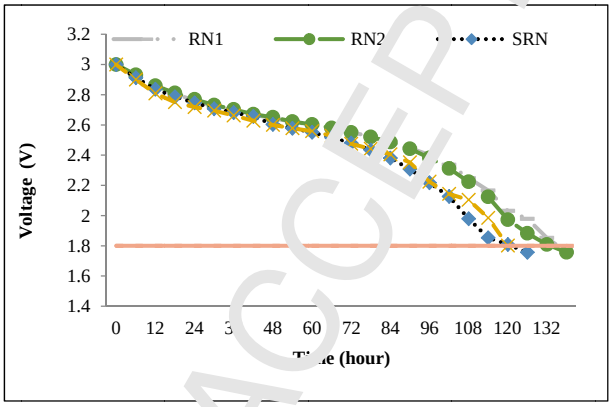

Figure 17 Network lifetime using 7 motes with AA energizer batteries

Table 6. Prototype end-to-end delay

delay	AVG	Max	Min	STD
Distributed RN	25.06838	34	1	5.07
Centralized RN	30.3836	221	1	15.82

In Figure 17, the network lifetime can be estimated for all types of nodes, e.g. RN1, RN2, SRN (single RN node), BS (Base Station). As shown in Figure 17, the base station and the single RN has the minimum network life time, which is around 120 hours, whereas the distributed node RN1 and RN2 have longer network time than the base station and the single RN. This is expected, since the four publishers are distributed over the two RNs, i.e. two publishers per each RN node. A very important observation is that the results are nearly the same, in opposite to the expected, since distributing the load would give nearly double the life time. The reason behind that, we used the TeolsB with its default state, which means the sensors are all the time in the receive mode, that makes the difference between all sensors quite small.

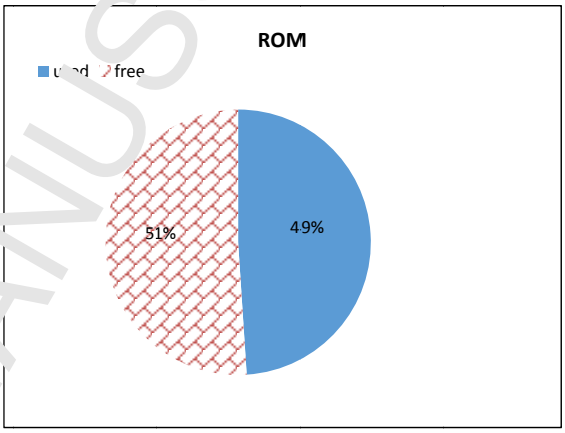

Figure 18. ROM occupied space after uploading EATDDS

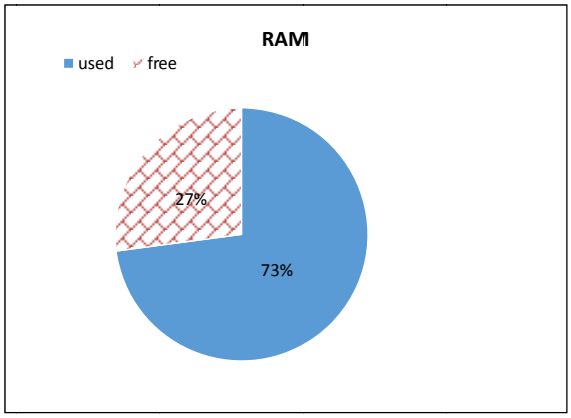

Figure 19. RAM occupied space after uploading EATDDS to TelosB

The memory is an important measure, specifically for the limited resources devices. It gives a clear evidence of the applicability of the developed technique. The memory measurements of EATDDS is shown in Figure 18 and Figure 19, for ROM and RAM respectively. The results show that the memory in both ROM and RAM still have free space around 51% and 27% for ROM and RAM respectively. In this regard, one important notice for TinyDDS memory is that increasing the number of subscribers increase the occupied memory significantly, in opposite to increasing the publishers.

6. CONCLUSION AND FUTURE WORK

In this paper, a novel energy-aware middleware protocol for WSAN was developed and named EATDDS. Since DDS is a middleware standard and currently it is widely used in academic and industry fields, it was selected to be the base of EATDDS protocol. Besides, a very important enhancement was added into TOSSIM simulator to make it capable of developing and simulating energy-aware mechanisms. Since EATDDS was tested over real sensors, we concluded that although WSAN has limited resources, still it can accommodate middleware and energy aware mechanisms. From the WSAN nature, the pub/sub communication model is the effective solution for such networks. Using pub/sub middleware to simplify the applications development and integration of distributed systems is on the cost of huge communication in the underlying layers. Therefore, it is not an easy task to adapt such solution for limited resources systems such as WSN. TinyDDS has several potential enhancements that can significantly reduce the overhead, such as using broker-less architecture. As an extension to this paper, we are continuing the improvement of TinyDDS to fit the restricted requirements of sensor-based networks. Also, QoS enhancements to EATDDS can be added, for example, reliability, deadline and priority QoS policies.

ACKNOWLEDGMENT

The authors would like to thank King Fahd University of Petroleum and Minerals for the support for this work.

REFERENCES

[1] OMG, "Data Distribution Services (DDS)," 1 1 2007. [Online]. Available: http://www.omg.org/spec/DDS/. [Accessed October 2016].

[2] RTI, "RTI Connext DDS," 2013. [Online]. Available: http://www.rti.com/products/dds/index.html. [Accessed October 2016].

[3] T. Sheltami, A. Al-Roubaiey and A. Mahmoud, "A survey on developing publish/subscribe middleware over wireless sensor/actuator networks," pp. 1-22, wireless networks, 2016.

[4] P. Boonma and J. Suzuki, "TinyDDS: an interoperable and configurable publish/subscribe middleware for wireless sensor networks," Handbook of Research on Advanced Distributed Event-based Systems, 2009.

[5] "TELOSB Crossbow," Crossbow, [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf. [Accessed 9 11 2016].

[6] P. Levis, N. Lee, M. Welsh and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys'03, 2003.

[7] TOSSIM, "TinyOS Documentation Wiki," 2003. [Online]. Available: http://docs.tinyos.net/index.php/TOSSIM. [Accessed 24 Nov. 2016].

[8] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer and D. Culler, "TinyOS: An Operating System for Sensor Networks," Ambient Intelligence, pp. 115-148, 2005.

[9] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva, "Directed diffusion for wireless sensor networking," IEEE/ACM Transactions on Networking, vol. 11, no. 1, pp. 2-16, 2003.

[10] E. Wang, Y. Ye and X. Xu, "Lightweight Secure Directed Diffusion for Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 2014, no. Article ID 415143, p. 12, 2014.

[11] U. Hunkeler, H. L. Truong and A. Stanford-Clark, "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks," in 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE 2008), 2008.

[12] A. Stanford-Clark and U. Hunkeler, "MQ Telemetry Transport (MQTT)," 1999. [Online]. Available: http://mqtt.org. [Accessed 22 9 2016].

[13] A. Stanford-Clark and H. L. Truong, "MQTT for sensor networks (MQTTs) specifications," IBM, Oct. 2007. [Online]. Available: http://www.mqtt.org/MQTTs. [Accessed 22 Sept. 2016].

[14] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss and P. Levis, "Collection tree protocol," in Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, 2009.

[15] J.-H. Hauer, V. Handziski, A. Köpke, A. Willig and A. Wolisz, "A Component Framework for Content-Based Publish/Subscribe in Sensor Networks," Wireless Sensor Networks Lecture Notes in Computer Science , vol. , no. 13, pp. 369-385, 2008.

[16] J. Chen, M. Díaz, L. Llopis, B. Rubio and J. M. Troya, "A survey on quality of service support in wireless sensor and actor networks: Requirements and challenges in the context of critical infrastructure protection," Journal of Network and Computer Applications, vol. 34, no. , pp. 1225-1239, July 2011.

[17] X. Tao and E. C. Ngai, "A Ubiquitous Publish/Subscribe Platform for Wireless Sensor Networks with Mobile Mules," in IEEE 8th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2012, 2012.

[18] J. Chen, M. Díaz, B. Rubio and J. M. Troya, "PS-QUASAR: A publish/subscribe QoS aware middleware for Wireless Sensor and Actor Networks," Journal of Systems and Software, vol. 86, no. 6, pp. 1650-1662, June 2013.

[19] A. S. Tanenbaum, Computer Networks, Boston: Prentice Hall PTR, 2011.

[20] Y. Tekin and O. K. Sahingoz, "A Publish/Subscribe messaging system for wireless sensor networks," in 2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), Konya, 2016.

[21] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in SenSys '04 Proceedings of the 2nd international conference on Embedded networked sensor systems, 2004.

[22] E. Perla, A. Cathain, R. Carbajo, M. Huggard and C. Goldrick, "PowerTOSSIM z: realistic energy modelling for wireless sensor network environments," in PM2HW2N '08 Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, 2008.

[23] M. Memsic, "MICAz," [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf. [Accessed 23 4 2016].

[24] D. Gay, P. Levis, D. Culler and E. Brewer, "nesC 1.2 Language Reference Manual," TinyOS, 2005.

[25] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, vol. 1, no. 4, pp. 660-670, 2002.

[26] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz and J. Kelner, "Mires: a publish/subscribe middleware for sensor networks," Personal and Ubiquitous Computing, vol. 10, no. 1, pp. 37-44, February 2006.

[27] P. Levis, N. Lee, M. Welsh and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in SenSys '03 Proceedings of the 1st international conference on Embedded networked sensor

systems, 2003.

[28] E. Perla, A. Catháin, R. Carbajo, M. Huggard and C. Goldrick, "PowerTOSSIM z: realistic energy modelling for wireless sensor network environments," in PM2HW2N '08 Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, 2008.

[29] Yasin Tekin, Ozgur Koray Sahingoz, "A Publish/Subscribe messaging system for wireless sensor networks," IEEE Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), pp. 171 – 176, 2016.

[30] G. Siegemund, V. Turau and K. Maâmra, "A self-stabilizing publish/subscribe middleware for wireless sensor networks," *2015 International Conference and Workshops on Networked Systems (NetSys)*, Cottbus, 2015, pp. 1-8.

[31] Yuvraj Sahni, Jiannong Cao, Xuefeng Liu, MidSHM: A Middleware for WSN-based SHM Application using Service-Oriented Architecture, Future Generation Computer Systems, Volume 80, 2018, Pages 263-274,

[32] Anas Al-Roubaiey, Tarek Sheltami, Ashraf Mahmoud, " ID-Based Routing Protocol for Wireless Network with a Grid Topology," US Patent Grant, US10149216B2, 2018.

Bio Data

Anas A. Al-Roubaiey

Anas A. Al-Roubaiey received the B.S. degree in computer engineering from Arab Academy for Science and Technology, Alexandrea, Egypt, in 2001, He worked as teaching assistant in Taiz University until 2004. The M.S. and Ph.D. degrees in computer networks from King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, in 2009 and 2015, respectively. He is currently a researcher engineer in King Fahd University. His research interests include intrusion detection systems, middleware

Tarek R. Sheltami

Tarek R. Sheltami received his Ph.D. in Electrical and Computer Engineering from the Electrical and Computer Engineering Department at Queen's University, Kingston, Ontario, Canada on April 2003. Dr. Sheltami is currently a Professor at the Computer Engineering Department at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Kingdom of Saudi Arabia. He joined the department on August 26, 2004. Before joining the KFUPM, Dr. Sheltami was a research associate professor at the School of Information Technology and Engineering (SITE), University of Ottawa, Ontario, Canada. He worked at GamaEng Inc. as a consultant on Wireless Networks (2002-2004). Also, he worked in several joined projects with Nortel Network Corporation. Dr. Sheltami has been a member of the technical program and organizing committees of several international IEEE conferences. His research interests include Ad hoc Networks, WSAN, IoT, Digitization, Computer Network Security and Performance Evaluation.

Ashraf Mahmoud

Ashraf S. Mahmoud: received the B.Sc. degree in Electrical and Computer Engineering from Kuwait University in 1990, the M.Eng. in Engineering Physics (Computer Systems) from McMaster University, Hamilton, Canada in 1992. He received his Ph.D. in Systems and Computer Engineering from Carleton University, Ottawa, Canada in 1997. During 1997-2002, he was with Nortel Networks Research and Development where he focused on development and evaluation of radio resource management algorithms for broadband and 3G networks. Since 2002, he is with the Computer Engineering department at King Fahd University of Petroleum and Minerals, Dhahran, Saudi. His research interests include performance evaluation and simulation techniques, mobile and wireless sensor networks, and IoT.

Ansar Yaser

Dr. Ansar Yasar is a professor at the Transportation Research Institute (IMOB) – Hasselt University Belgium. IMOB is the one of the top European institutes focusing on the broader themes of road safety and transport management. At IMOB, he worked on the European FP7 project DATA SIM (2011 - 2014) & European ERA-NET Smart-PT (2014 - 2017). He is currently responsible for the H202-Track&Know (2018-2020), & H2020 iSCAPE (2016 - 2019) projects with a consortium of several international partners. He is responsible for the Intelligent Transport Systems (ITS) and Intelligent Topics in Transportation courses for the masters of transportation sciences students & related research around this topic at UHasselt. Furthermore, he is currently the Head of the Business Development Unit at the institute focusing on several applications including Safe Routes to Schools & Safe-Trucking System. Dr. Yasar's broader research interests include smart cities & communities, connected & intelligent mobility, drones system management, road safety using smart transport solutions and mobility management. He received his BS degree in Software Engineering from Foundation University Islamabad, MS degree in Computer Science & Engineering from Linkoping University - Sweden and a PhD in Engineering in from Katholieke Universiteit Leuven - Belgium. He has authored more than 80 research articles in renowned international journals, conferences and

workshops. As one of his accomplishments, Dr. Yasar has co-edited a book entitled “Data Science and Simulation in Transportation Research” published by IGI Global in December 2013. Furthermore, he has been involved in organization of many international peer-reviewed conferences, summer schools and other scientific events. Dr. Yasar is also a Technical Expert to evaluate project proposals submitted to the European R&D – EUREKA & COST frameworks.

Anas A. Al-Roubaiey

Tarek R. Sheltami

Ashraf Mahmoud

Ansar Yaser

Here are some highlights about the paper:

1. We have implemented an effective energy-aware middleware for WSAN based on DDS standard, which is called EATDDS.
2. We also developed an online Energy Model (OEM) to make TOSSIM capable of developing and testing energy-aware protocols
3. The model is validated by comparing it against POWERTOSSIM.
4. Our results show that EATDDS is efficient and can be accommodated with limited system resources.