

**ENERGY-AWARE PUBLISH/SUBSCRIBE DDS-BASED
MIDDLEWARE FOR WIRELESS SENSOR AND ACTUATOR
NETWORKS**

BY

ANAS ABDELWAHID AL-ROUBAIEY

A Dissertation Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

COMPUTER SCIENCE AND ENGINEERING

MAY 2015

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

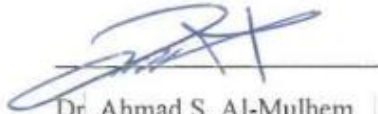
DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

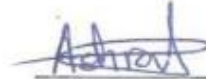
This thesis, written by Anas Abdelwahid Al-Roubaiey under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING.**



Dr. Tarek Sheltami
(Advisor)



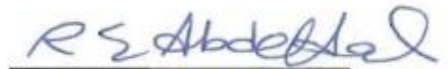
Dr. Ahmad S. Al-Mulhem
Department Chairman



Dr. Ashraf S. Mahmoud
(Co-Advisor)



Dr. Salam A. Zummo
Dean of Graduate Studies



Dr. Radwan E. Abdel-Aal
(Member)

30/7/2015
Date



Dr. Shokri Selim
(Member)



Dr. Moataz Ahmad (Member)

© Anas Al-Roubaiey

2015

أهدي هذا العمل المتواضع إلى والديّ، حفظهما الله ورعاهما |

To my parents, may Allah protect them and give them good health and happiness |

ACKNOWLEDGMENTS

الحمد لله الذي بنعمته تتم الصالحات، والحمد لله من قبل ومن بعد على توفيقه وعونه.

Praise be to Allah , Lord of all creation

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	V
TABLE OF CONTENTS.....	VI
LIST OF TABLES.....	IX
LIST OF FIGURES.....	X
LIST OF ABBREVIATIONS.....	XIII
ABSTRACT.....	XVI
ملخص الرسالة.....	XVII
1 CHAPTER INTRODUCTION	1
1.1 Problem Statement.....	7
1.2 Proposed Solution	9
1.3 Research Questions	11
1.4 Research Objectives.....	12
1.5 Dissertation Organization	13
2 CHAPTER LITERATURE REVIEW	15
2.1 Pub/Sub Model Overview.....	16
2.1.1 Principles of The Pub/Sub Model.....	17
2.1.2 Pub/Sub Middleware Components	19
2.2 Pub/Sub in WSAN	34
2.2.1 Existing Solutions	34
2.2.2 WSAN Pub/Sub Reference Model	45

3	CHAPTER A PUB/SUB MIDDLEWARE COST IN SENSOR NETWORKS	50
3.1	Case Study.....	50
3.2	Cost and Performance Evaluation	53
4	CHAPTER RTDDS: RELIABILITY PROTOCOL FOR WSAN.....	59
4.1	RTDDS Implementation.....	59
4.2	Performance Evaluation.....	66
4.2.1	Experiments setup.....	68
4.2.2	Full Reliability QoS Results	70
4.2.3	Partial Reliability QoS Results	75
4.2.4	Memory and Energy Consumption Results.....	78
5	CHAPTER ONLINE ENERGY MODEL	81
5.1	Online Energy Model Description.....	82
5.1.1	Radio Component	83
5.1.2	Microcontroller (MCU) Component	85
5.2	Simulation and Validation.....	87
6	CHAPTER BROCKER-LESS TINYDDS.....	90
6.1	Proposed Solutions	90
6.1.1	Default TinyDDS	91
6.1.2	Broker-Less TinyDDS.....	92
6.1.3	Hybrid TinyDDS	93
6.2	Performance Evaluation.....	95
6.2.1	Test Scenario and simulation setup	95
6.2.2	Performance Metrics.....	97
6.2.3	Results and Analysis	99

7	CHAPTER EATDDS.....	114
7.1	EATDDS Description.....	114
7.2	Performance Evaluation.....	116
7.2.1	Experiment setup.....	116
7.2.2	Performance metrics.....	117
7.2.3	Results and analysis.....	118
8	CHAPTER CONCLUSIONS AND FUTURE WORKS.....	126
	REFERENCES.....	130
	APPENDIX A.....	142
	ID-Based Routing.....	142
	APPENDIX B.....	146
	Adaptive Reliability Protocol for Wireless Sensor Networks using Packet Delivery Ratio Metric.....	146
	VITAE.....	154

LIST OF TABLES

Table 1 Pub/Sub WSAN Solutions (D: Deadline; P: Priority; R: Reliability).....	47
Table 2 Pub/Sub WSAN Solutions, Evolution and Features Summary	48
Table 3 Simulators Used in Evaluating Pub/Sub Solutions for WSAN	49
Table 4 Simulation setup	69
Table 5 The three main scenarios in the simulation study	69
Table 6 RTDDS performance with high and low data rates	75
Table 7 Radio Current Consumption of MicaZ and TelosB.....	85
Table 8 MCU Current Consumption of MicaZ and TelosB	87
Table 9 The OEM and PTZ validation comparison.....	89
Table 10 Simulation setup	97
Table 11 MicaZ Energy model parameters.....	105
Table 12 Energy consumption figures' symbols	110
Table 13 Prototype end-to-end delay	123

LIST OF FIGURES

Figure 1 Traditional WSN architecture.....	2
Figure 2 WSN architecture with partially and fully automated interaction.....	3
Figure 3 Middleware layer hides the complexity of underlying layers	7
Figure 4. Simple architecture for publish/subscribe communication model	18
Figure 5. Main components of publish/subscribe middleware	20
Figure 6. The 3-Layers WSN architecture	22
Figure 7. General message format	23
Figure 8. Pub/Sub schemes	24
Figure 9. Topics represented by C++ structures	26
Figure 10. Content-based vs. Topic-based Pub/Sub interaction.....	28
Figure 11. Notification Service.....	30
Figure 12. Directed Diffusion simplified schematic. (a) Interests propagation. (b) Gradients setup. (c) Reinforced path.	36
Figure 13. Mires Architecture.....	37
Figure 14. MQTT-S architecture	38
Figure 15. TinyCOPS architecture.....	39
Figure 16. PS-QUASAR architecture	40
Figure 17. UPSWSN-MM publish/subscribe system	41
Figure 18. TinyDDS architecture over TinyOS and MicaZ platform.....	44
Figure 19 Pub/sub middleware reference model	46
Figure 20: Case study network topology; Sub (BS): subscriber (base station), Pub (Sr): publisher (sender).	52
Figure 21. Basic application algorithm.....	52
Figure 22 Packet delivery ratio comparison.	54
Figure 23 End-to-end delay comparison.....	56
Figure 24 Memory cost comparison.	57
Figure 25 Energy consumption comparison	58
Figure 26 RTDDS Architecture	62
Figure 27 RTDDS Classifier on the publisher side	64
Figure 28 The reliability algorithm of RTDDS with FRQoS level	65

Figure 29 PRQoS level algorithm.....	66
Figure 30 Interference effect on the performance of RTDDS	71
Figure 31 The impact of RTO and No. of hops on RTDDS performace.....	72
Figure 32 The impact of number of publishers and data rate on RTDDS performance...	74
Figure 33 Partial Reliability QoS results with five and one seconds IPI and five publishers	76
Figure 34 RTDDS and TinyDDS Memory consumption based on TelosB platform.....	79
Figure 35 Energy consumption of RTDDS (FRQoS level) and TinyDDS.....	80
Figure 36 Online Energy Model Architecture	83
Figure 37 Energy consumption of the Radio Component	88
Figure 38 Energy Consumption of the MCU component.....	89
Figure 39 The sequence diagram of the discovery and data dissemination phases of DefTDDS	92
Figure 40 The sequence diagram of the discovery and data dissemination phases of BLTDDS	93
Figure 41 The sequence diagram of the discovery and data dissemination phases of HyTDDS	94
Figure 42 The tested scenario with 5 publishers, 3 subscribers and 3 topics	96
Figure 43 The number of discovery process messages vs. IPI	101
Figure 44 The average discovery time of new subscriber	101
Figure 45 Packet delivery ratio of the reliable scenario	102
Figure 46 Packet delivery ratio of the best-effort scenario.....	103
Figure 47 End-to-End scenario of the reliable scenario.....	104
Figure 48 End-to-End delay of the best-effort scenario.....	104
Figure 49 Total energy consumption in milli-Joule for reliable QoS.....	106
Figure 50 Total Energy Consumption in milli-Joule for Best Effort QoS.....	106
Figure 51 Radio and MCU energy consumption of DefTDDS with Reliable QoS.....	108
Figure 52 Radio and MCU energy consumption of BLTDDS with Reliable QoS.....	108
Figure 53 Radio and MCU energy consumption of HyTDDS with Reliable QoS.....	109
Figure 54 DefTDDS energy consumption distribution over the network nodes	111
Figure 55 BLTDDS energy consumption distribution over the network nodes	113

Figure 56 HyTDDS energy consumption distribution over the network nodes	113
Figure 57 Cluster formation of EATDDS.....	115
Figure 58 The network Total Energy Consumption	118
Figure 59 Remaining energy at the end of network life time	119
Figure 60 Packets per Joule vs. Inter-Packet Interval.....	119
Figure 61 Network life time at the moment the first node dies	120
Figure 62 TelosB mote platform.....	121
Figure 63 Experiment environment and testbed	122
Figure 64 The Base Station attached to the PC USB port	122
Figure 65 Network life time using 7 motes with AA energizer batteries	123
Figure 66 ROM occupied space after uploading EATDDS.....	124
Figure 67 RAM occupied space after uploading EATDDS to TelosB mote	125
Figure 68: Id-based Routing method	143
Figure 69 Rounds distribution over the network life time	150
Figure 70 Round algorithm flowchart.....	151
Figure 71 Adaptive reliability switching algorithm flowchart	152
Figure 72 Adaptive reliability regions	153

LIST OF ABBREVIATIONS

BE	:	Best-Effort
BEQoS	:	Best-Effort Quality of Service
BLTDDS	:	Broker-Less TinyDDS
BLTDDS	:	Broker Less TinyDDS
DDS	:	Data Distribution Service
DefTDDS	:	Default TinyDDS
DMR	:	Dropped Message Ratio
DYMO	:	Dynamic MANET On demand
EATDDS	:	Energy-Aware TinyDDS
EDF	:	Earliest Deadline First
EED	:	End-to-End Delay
FIFO	:	First In First Out
FR	:	Full Reliability
FRQoS	:	Full Reliability Quality of Service
HyTDDS	:	Hybrid TinyDDS
HyTDDS	:	Hybrid TinyDDS
IFS	:	Interference Free Scheduling

IPI	:	Inter-Packet-Interval
JMS	:	Java Message Server
MANET	:	Mobile Ad-hoc NETworks
MCU	:	MicroController Unit
PDR	:	Packet Delivery Ratio
PR	:	Partial Reliability
PRQoS	:	Partial Reliability Quality of Service
Pub/Sub	:	Publisher/Subscriber
QoS	:	Quality of Service
RAM	:	Random Access Memory
Rd/Msg	:	Redundant per Message
R-DR	:	Reliable Data Reader
R-DW	:	Reliable Data Writer
ReTx	:	Retransmissions
RN	:	Rendezvous Node
ROM	:	Read Only Memory
RTDDS	:	Reliable TinyDDS

RTO : Retransmission TimeOut

SA : Sensor/Actuator

TOSSIM : TinyOS SIMulator

WSAN : Wireless Sensor/Actuator Network

WSN : Wireless Sensor Network

|

ABSTRACT

Full Name : [Anas Abdelwahid Al-Roubaiey]
Thesis Title : [ENERGY-AWARE PUBLISH/SUBSCRIBE DDS-BASED
MIDDLEWARE FOR WIRELESS SENSOR AND ACTUATOR
NETWORKS]
Major Field : [Computer Science and Engineering]
Date of Degree : [May, 2015]

In the recent years, the publish/subscribe (pub/sub) communication model has emerged as a suitable communication paradigm for large-scale distributed systems. That is due to its effective decoupling properties for the network's participants in time, space, and synchronization. These properties are well-suited for Wireless Sensor/Actuator Networks (WSAN) applications. Data Distribution Service (DDS) is a well-known standard in the academic and industrial communities for supporting real-time distributed systems based on the pub/sub model. In addition to the pub/sub model advantages, DDS has a rich set of Quality of Service (QoS) policies. Therefore, porting DDS function into WSAN may significantly improve its performance in terms of QoS support, scalability, portability, and interoperability. TinyDDS is a light weight and partial porting of DDS middleware to WSN platforms. As such, TinyDDS in its current form has several limitations, such as: (1) it does not support any form of reliability for data delivery, (2) it has a battle neck problem in the event routing protocol, and (3) it does not have an energy aware mechanism to tackle the scarce energy source problem of WSAN. This work added several contributions to the efforts of porting DDS standard benefits into WSAN. These contributions are: (1) a comprehensive review for the pub/sub model, and the state of the art solutions of integrating pub/sub into WSAN is conducted; (2) The cost of adding pub/sub model into WSAN is thoroughly evaluated; (3) the DDS reliability QoS is improved to suit WSAN requirements and the improved DDS reliability QoS is ported into TinyDDS; (4) the problem of central event routing is tackled by proposing broker-less solutions; (5) an energy aware protocol is developed and tested.

ملخص الرسالة

الاسم الكامل: أنس عبد الواحد حسن الربيعي

عنوان الرسالة: برنامج النشر والإشتراك لشبكات الإستشعار والتحكم بموثوقية نشر البيانات ومعلومات الطاقة

التخصص: علوم وهندسة الحاسب الالى

تاريخ الدرجة العلمية: مايو، 2015

في السنوات الأخيرة، برز نموذج النشر والإشتراك (نشر/إشتراك pub/sub) كنموذج إتصالات مناسب جدا للنظم الموزعة على نطاق واسع. ويعود ذلك إلى خصائصه الفعالة في عدم ربط النظام بوقت الإرسال، عناوين المرسلين، وتزامن نقل البيانات. هذه الخصائص تعتبر مناسبة تماما لشبكات الاستشعار والتحكم اللاسلكية (WSAN). مؤخرًا، برز ايضا معيار معتمد لنموذج النشر والإشتراك، يسمى خدمة توزيع البيانات (DDS)، وهو معيار مشهور و معروف جدا اكاديميا و صناعيا. يتمتع هذا المعيار بالإضافة إلى خصائص النشر والإشتراك بأنه يحتوي على مجموعة غنية من سياسات جودة الخدمة (QoS) التي تطور أداء الأنظمة الموزعة بشكل فعال وكفاءة عالية. وكجهد اولي لنقل مميزات DDS إلى شبكات الاستشعار، والتي تعاني بشكل كبير من نقص الموارد كالمعالجة والتخزين والطاقة، قامت مجموعة خبراء من جامعة ماساتشوستس في بوسطن بتطوير نسخة مصغرة من DDS لتتناسب قدرات الشبكات الاستشعارية، وسمية هذه النسخة بتقنية TinyDDS النسخة المصغرة من DDS. وكأي جهد اولي، لا زال هناك بعض الثغرات في TinyDDS والتي يمكن ايجازها فيما يلي: (1) لا يدعم أي شكل من اشكال الموثوقية لنقل البيانات. (2) يستخدم النقل المركزي للبيانات والذي لا يتناسب اطلاقا مع القدرات المحدودة لشبكات الإستشعار. (3) لا تعتمد وظيفته على توزيع الطاقة المتبقية على عناصر الشبكة مما قد ينهي عمر الشبكة ولا زال لديها كمية كبيرة من الطاقة المخزنة. في هذا العمل، تقدم نظرة شاملة لهذه التقنيات ونضع حلولاً لمشاكل TinyDDS مع اختبار دقيق ومقارنة عادلة للحلول المقدمة مع النسخة الاصلية ل TinyDDS. ويمكن حصر المساهمات التي يقدمها هذا العمل كالتالي: (1) مراجعة شاملة لنموذج ال نشر/إشتراك، ودراسة دقيقة للحلول المقدمة طبقا لهذا النموذج لشبكات الاستشعار مع مقارنتها مع بعضها من حيث المميزات والعيوب. (2) تقديم دراسة وتحقيق متكامل عن التكلفة الفعلية لإضافة هذا النموذج، متمثلا بال DDS، إلى الشبكات الإستشعارية المحدودة القدرات. (3) تحسين جودة الخدمة والموثوقية في نقل البيانات للمعيار DDS لتتناسب قدرات الشبكات الاستشعارية مع تطوير ونقل هذه الخدمة إلى TinyDDS. (4) حل المشكلة المركزية في نقل البيانات في تقنية TinyDDS بتقديم حلول تنهي تماما أو جزئيا هذه المشكلة. (5) واخيرا، تقديم حل متكامل لكل ما سبق مع تطوير أداء TinyDDS بإضافة تقنية تعتمد في نقل البيانات على مراقبة الطاقة المتبقية في عناصر الشبكات اللاسلكية مما ينتج عنه زيادة في عمر الشبكات الاستشعارية للمراقبة والتحكم (WSAN).

CHAPTER 1

INTRODUCTION

Sensor networks are composed of tens/hundreds of low-priced and tiny devices with limited capabilities that are deployed to an area of interest to monitor the behavior of a particular phenomenon. In traditional single sink/base station WSN applications, the data flow usually moves from the sensors to the monitoring application through a sink node, as shown in Figure 1. The deployed sensors collect and send the data to the sink node using one-to-many communication pattern [1]. Thus, the main function on WSN was to sense and collect the data from the surrounding area without doing any action. Many applications benefit from this functionality, such as environmental monitoring, Structural Health Monitoring (SHM), Human Health Monitoring (HHM), habitat monitoring, and military surveillance. However, due to the recent advances in sensor-based network technology the Wireless Sensor and Actor Networks (WSAN) have emerged as enabling technology for in-network decision making, where the network can sense and react without the need to go to external and control applications [2].

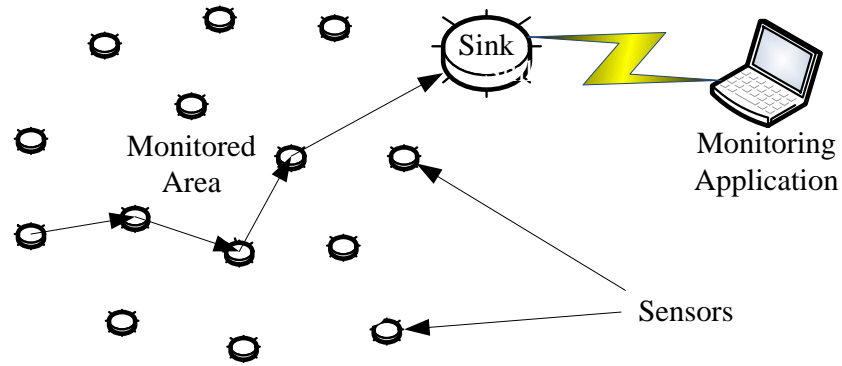


Figure 1 Traditional WSN architecture

Figure 2 depicts this technology, and how it supports the process automation in many applications such as home automation, Industrial Process Automation (IPA), detection and reaction systems, e.g., nuclear, chemical and toxic gas attacks detection, and recently smart cities and Internet of Things (IoT); where WSN is one of the main enabling factors of IoT [3] [4]. According to the data exchange in WSN, the process automation in WSN applications can be classified into partial and fully automated applications [5].

Figure 2, part (a) illustrates the partial automation interaction, where the sink is involved in decision making, which is more centralized and controlled, but incurs more delay. In contrast, in the fully automated interaction, as shown in

Figure 2 part (b), the sensors sense the data and send it directly to the actuators for processing and reacting in response to the result of the local data analysis. The fully automated approach is more suitable for real-time applications since it reduces the time and overhead of centralized approach, i.e. partial automated approach.

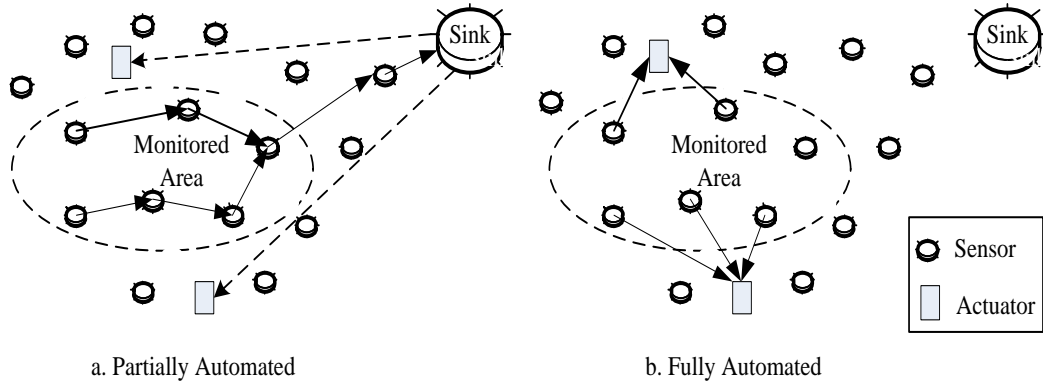


Figure 2 WSN architecture with partially and fully automated interaction

After knowing what WSN is, we define the publish/subscribe model and its suitability for WSN. A publish/subscribe (pub/sub) paradigm is a messaging based communication model, where senders, called publishers, send their data to a logical data space, called middleware, without knowledge of who or where are the receivers, called subscribers. Similarly, subscribers receive only the data of interest, without knowledge of who or where are the publishers. The Pub/Sub interaction paradigm is designed to suite large-scale distributed real-time applications. Oh et al. [6] have done a suitability analysis for pub/sub scheme, their main remarks were as follows:

- Pub/sub model has advantage when system is large and data transfer is shared among many clients; which is mostly the case in sensor networks where large number of sensors are deployed to deliver the monitored object information to multiple sinks and/or actuators.
- Pub/sub model is suitable when events or data updates occur infrequently. For example, event-based applications that mainly monitor and control distributed systems (e.g. WSN).

- Pub/sub model is suitable when the degree of common interest is high. For example, in WSN applications the data gathered by sensors highly has a common interest by the multiple sinks, applications, or actuators.
- Pub/sub model is more suitable than request/replay model in less user intervention applications.
- In pub/sub model data updates are immediately delivered to subscribers which is more suitable for real-time applications where deadline is short or strict. For example, in battlefield surveillance WSN.
- When clients seldom use published data, pub/sub model is not suitable.

The scalability and robustness of the paradigm came from its decoupling properties in time, space, and synchronization [7]. Particularly, these properties make it more suitable for data-centric sensor network applications. Moreover, the sensor network applications have distinct characteristics that make Pub/Sub middleware the appropriate solution for such environments [8] [9].

These main characteristics and design issues that make pub/sub suitable for WSN are as follows:

Many-to-Many Interaction. Multiple sinks (base stations) sensor networks and WSN applications migrate the sensor network based applications from one-to-many to many-to-many communication model. In these new applications, the data should flow in both directions from sensors to actuators or sinks and vice versa. For example, the main role of the sensor is to publish the data that is collected for the monitored area, whereas the actuator

is the subscriber who subscribe to the sensor data to be analyzed and do some appropriate action. However, the sensor also needs to be a subscriber to get the control data from the sink (e.g. sleep, wake up, or configuration data like software new setting parameters and updates); also the actuator needs to be a publisher to send the information to the sink nodes. Thus, the large-scale distributed sensors and actuators with many-to-many communication requirements are realized by Pub/Sub interaction model, where it is basically a many-to-many communication model [7].

Data-Centric. Data-centricity is a key feature of WSNs that distinguishes them from other wireless data networks; it provides efficient usage of their limited resources and matches well their nature [10]. In WSN the application is not interest in the identity of the sensor, rather the interest is in the data gathered from the monitored physical environment. Nowadays many applications may be interested in different types of data from the same WSN infrastructure, whereas the traditional single sink WSN applications were designed mostly to support one application per network. For example, a building monitoring application may need to concurrently monitor the building temperature, wall cracks, light intensity, and movements. Moreover, it may contain actuators to support physical reaction, e.g. reduce the building temperature by loosening the cooling valve in the cooling system. This type of applications leads to the concept of data-centric producer/consumer (Pub/Sub) communication paradigm; where the subscribers (sinks, actuators and end user applications) are interested in the information coming from the publishers (sensors), and they do not know exactly from where the data comes in terms of network address. Another example, in a tracking system the end-user who is responsible for the monitoring process is interested in the location of the monitored object, but not in

the addresses of the GPS devices which delivered this information [11]. From the data flow of the sensor device, it can be a publisher, e.g. publishing readings like temperature, gas intensity, location, or humidity, and a subscriber at the same time, e.g. subscribing to controls signals.

Network Dynamics. Although sensor networks are mostly stationary, the dynamicity appears in several situations; (1) when the nodes are joining or leaving the network due to the hardware or software failures in the node or network links (wireless links are error-prone). (2) New applications may be added at the end-user monitors or crashed due to software errors. (3) For energy saving, node state changes continuously from active to sleep modes or deep sleep modes where it may join again with new network address. (4) Some of the WSN network protocols are changing their network addresses from time to time (e.g. ZigBee) [12]. This dynamic network behavior makes Pub/Sub interaction paradigm the most suitable solution for such type of networks. In which the data is stored in buffers (queue structures) and submitted whenever there is a connection (decoupling in time property). Moreover, the Pub/Sub middleware hides the underlying network details from the application to mitigate the network addresses continuous changes when nodes leave and join the network.

Heterogeneity. Currently different varieties of sensor platforms exist in the industry field, due to the lack of standards in WSN technology [13]. As a result, a tightly coupled application is developed to meet the applications' requirements, where the developers should be aware of the detailed information of underlying network layers of the targeted platform. Also, it will be a very difficult and complex task when they want to integrate different platforms or integrate the WSN to the pub/sub-based enterprise networks [11].

Eventually, after intensive efforts that have been done by developers, a tightly coupled complex applications are developed, where these applications are very complex, not reusable, not portable, and even very difficult to upgrad. For instance, *Valley Forge* ship sank on 2 November 2006 because the system software could not integrate new technology and modern weapons; shockingly, upgrading the software cost too much to justify the existence of a billion-dollar asset [14]. The pub/sub middleware comes to mitigate this problem by implementing an intermediate layer between applications and underlying platforms ,as shown in Figure 3, to ease the applications development and makes them more portable, interoperable, and upgradable.

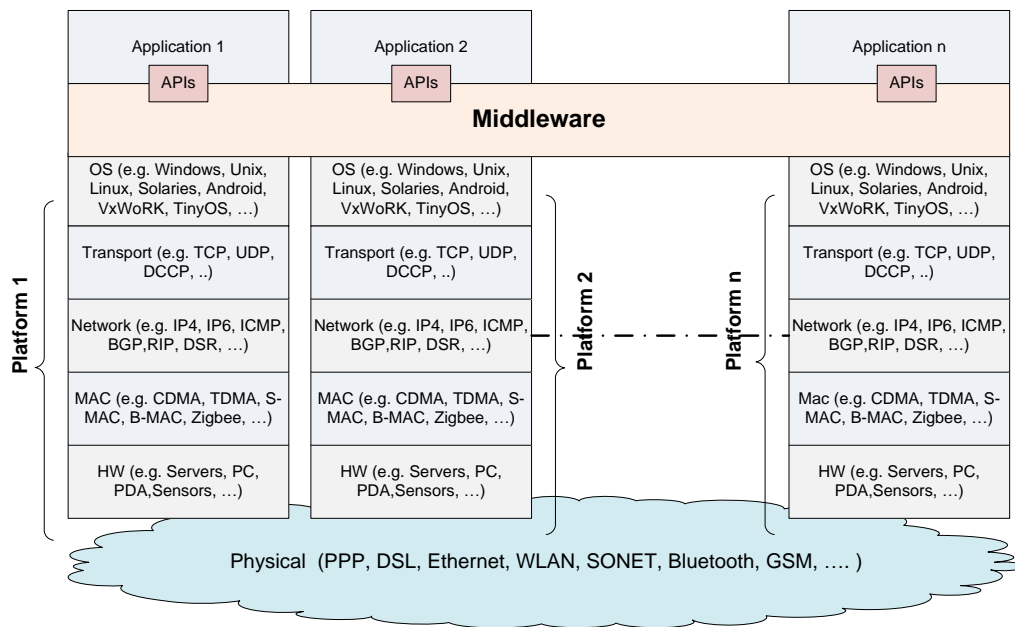


Figure 3 Middleware layer hides the complexity of underlying layers

1.1 Problem Statement

The data-centricity and decoupling properties in time, space, and synchronization of the pub/sub interaction scheme make it an appropriate solution for real-time and large-scale distributed computing systems. Also, Data-centricity and decoupling properties are key

features of WSNs that distinguish them from other wireless data networks. Therefore, the pub/sub model provides efficient usage of their limited resources and matches well their functionality [10] [15] [16]. Many works have been done in enabling publish/subscribe interaction scheme in WSN/WSNs [17] [18] [16]. However, to the best of our knowledge, none of them thoroughly investigated the cost of enabling this technology in networks with constrained resources like WSN; most of their focus was on adapting the pub/sub service for sensor networks and providing them with Quality of Service (QoS) support.

With limited resources networks, it is important to evaluate the cost of adding an advantage. The major resource constraint in WSN is energy, where sensors and actuators are battery-powered devices. In most cases, it would be very costly and difficult, impossible sometimes, to change their batteries due to the hazard and harsh environments where they are deployed, e.g. in battle field surveillance. Therefore, energy saving is a critical issue in such type of networks, where it highly impacts network life time. Unfortunately, most of the pub/sub WSN/WSN proposed solutions have not been evaluated in terms of energy consumption; consequently, nearly no energy saving techniques have been proposed. One recently proposed solution [19] has taken this in account and added energy consumption balancing technique in his proposed middleware. However, the energy consumption evaluation and analysis was very brief and did not even evaluate the QoS parameters cost in terms of energy consumption. Moreover, the proposed solution was not standard-based solution that would facilitate the integration of sensor/actuator networks to enterprise networks.

1.2 Proposed Solution

To tackle the lack of standardization and QoS support in WSN/WSAN, our proposed solution is, unlike the previous work, standard-based middleware, based on Object Management Group (OMG) Data Distribution Service (DDS), and focuses on evaluating the actual cost of applying pub/sub interaction scheme to WSAN; specifically in terms of energy consumption, memory footprint, and communication overhead. Furthermore, improvements are added to the existing pub/sub WSAN protocol to get an energy aware protocol and to fit the WSAN. Here, the big question would be *why we selected DDS-based solution?*, the answer is in the following points:

- DDS is a well-known pub/sub middleware standard and widely used in current enterprise networks, which should facilitate the integration of such networks to WSANs.
 - “DDS proposes an interesting pub/sub abstraction that greatly simplifies the communication tasks. It also provides a way to specify QoS constraints in the communication and the fact that it is an OMG standard makes it an attractive option for WSAN CIP (Critical infrastructure protection) systems” [20].
- That makes it a potential unified middleware for WSAN.
- DDS provides a rich of QoS policies that can be ported into WSAN
- Energy consumption for DDS-based solutions have not been thoroughly investigated yet in the WSN/WSAN context.
- No energy saving or balancing mechanism has been proposed.

- WSN not evaluated yet, where the only evaluated DDS-based solution was for WSN.

In an attempt to integrate the various benefits in terms of standardization, real-time communication and QoS functions into WSN, TinyDDS [21] provides a light-weight and partial porting of the DDS middleware for WSN. Unfortunately, the current porting of the TinyDDS [22] has the following limitations:

- It does not thoroughly investigate the TinyDDS cost
- It does not implement any of the reliable data delivery functions supported in the original DDS standard.
- It still has a centralized problem, which leads to bottleneck and single point of failure
- It does not have an energy aware mechanism

In this work, all these limitations are tackled and a final energy-aware version that provides the pub/sub and DDS technologies benefits is developed. Specifically, a thorough cost evaluation for adding TinyDDS to sensor-based networks is conducted. Throughout the remainder of this work, we refer to both WSN and WSN as *sensor-based networks*. In adding reliability to TinyDDS, we enhance the original DDS standard middleware by adding a third reliable data delivery level, referred to here by partial reliability (PR), and port all the respective reliability functions into the existing TinyDDS. The resulting new middleware is referred to by Reliable-TinyDDS (RTDDS). For the centralized problem, we proposed two main solutions that totally eliminate the central control node in TinyDDS. The new solutions are extensively evaluated via simulations and prototyping. The last

enhancement is the addition of an energy-aware mechanism to RTDDS, to minimize the energy consumption and thus prolong network life time. The resulting new middleware is referred to as Energy-Aware TinyDDS (EATDDS).

1.3 Research Questions

RQ1: What are the existing techniques of applying pub/sub model over sensor-based networks? What are their capabilities and limitations?

RQ2: What is the actual cost of applying pub/sub model on sensor-based networks in terms of memory footprint, energy consumption, and communication overhead?

RQ3: What are the main limitations of TinyDDS?

RQ4: To what extent would the pub/sub middleware scale up, in terms of number of nodes and/or work load, specifically when using full reliability QoS?

RQ5: What is the efficient energy consumption model that can be developed to test the default TinyDDS and its enhanced versions in terms of energy consumptions?

RQ6: What are the possible improvements to be added to TinyDDS to get a low energy consumption protocol while continuing to support QoS?

RQ7: What are energy saving mechanisms that can improve the performance of TinyDDS and prolong the network life time?

1.4 Research Objectives

The main objective of this research is to use standard-based solution to minimize the energy consumption of WSANs when applying the pub/sub interaction scheme, while maintaining the QoS support. Specifically, the objectives of this study are as follows:

1. Conducting an extensive literature review for the following:
 - The pub/sub interaction scheme (i.e. its main concepts, components, architectures, and variants).
 - Previous work in pub/sub middleware for WSN and WSAN, and conducting a comparison study.
2. Conducting extensive simulations to thoroughly investigate the following issues:
 - Energy and memory consumptions in different scenarios and under different workloads with/without using pub/sub interaction scheme.
 - Network performance in terms of packet delivery ratio and end-to-end delay with/without applying pub/sub middleware.
3. Developing an energy consumption model that can be used to estimate the default and enhanced versions of TinyDDS, and to develop EATDDS. Note that this objective is due to the lack of online energy consumption measurements in TinyOS simulators, which restricts the TinyOS research community from developing energy-aware protocols for TinyOS based applications.

4. Implementing and testing a reliability protocol for TinyDDS, by improving and porting the reliability QoS levels of DDS standard, called RTDDS.
5. Investigating the different solutions that can be added to TinyDDS middleware to come up with a reliable and fully decentralized middleware.
6. Developing an energy-aware protocol for TinyDDS that improve its energy consumption and prolong network life time, called EATDDS.

1.5 Dissertation Organization

The remaining part of this thesis is organized as follows:

- Chapter 2: introduces a comprehensive background about pub/sub communication model, including its main properties, functions, components. In addition, this chapter surveys and compares the state of the art solutions of the pub/sub middleware for sensor-based networks. Finally, this chapter introduces a general reference model of pub/sub middleware for WSAN.
- Chapter 3: the main point of this chapter is evaluating the cost evaluation of integrating pub/sub middleware into sensor-based networks. It provides a description of the scenarios used in the evaluation study, and presents the results and analysis.
- Chapter 4: introduces the first TinyDDS improvement, where the reliability protocol RTDDS' implementation is described in details. It includes the porting of DDS standard reliability QoS levels into TinyDDS, and also the new reliability QoS level that we added to meet the requirements of WSAN applications. Also, this chapter includes the simulation experiments, results,

and analysis of RTDDS. In addition, a prototype of RTDDS results and analysis are described.

- Chapter 5: introduces the TinyOS SIMulator (TOSSIM) enhancement that enable it to measure the energy consumption metric. It describes in detail the online energy model and its implementation in TOSSIM simulator. Also, it includes the validation study of this model by comparing its results with a well-known simulator in this field.
- Chapter 6: this chapter uses the model introduced in chapter 5 to evaluate the proposed solutions of the single point of failure problem in TinyDDS. Where two techniques are proposed: (1)The Broker-Less that completely eliminates the TinyDDS broker, and (2) the Hybrid solution that partially eliminates the TinyDDS broker. It also includes the simulation results and analysis.
- Chapter 7: this chapter describes improvements introduced to RTDDS to make it an energy-aware protocol, and the final version is called EATDDS. It includes the EATDDS protocol description and also the comparative study of RTDDS and EATDDS results and analysis.
- Chapter 8: introduces conclusions, recommendations. |

CHAPTER 2

LITERATURE REVIEW

Many studies have been conducted to adapt Pub/Sub communication model to WSN. In this paper, we review the Pub/Sub interaction paradigm in the context of WSN. Moreover, we classify, analyze and synthesize different solutions proposed recently in WSN/WSAN and discuss the open problems and new research directions in the area. Finally, we propose a new reference model for pub/sub middleware in wireless sensor and actuator networks. To the best of our knowledge this is the first survey on pub/sub in WSAN in the literature.

Pub/Sub interaction scheme has been proven as a scalable and robust solution in many applications, including many industrial systems and research prototypes. Several surveys have been conducted in the literature on pub/sub systems and prototypes [7] [23] [24] [25]. However these surveys were general and not specific for WSN/WSAN and limited resources systems; where the focus on sensor-based networks was very little. Also, numerous previous surveys were more generic under the title of middleware in WSN [26] [27] [28]. In contrast, this study is more specific where it is totally focused on the Pub/Sub solutions and covers state of the art solutions. These solutions are thoroughly described, investigated, and compared in detail in terms of their architectures, implementations, and evaluation mechanisms.

The methodology used to perform the conducted searches is described in this paragraph. Our concern was mainly on papers published during the last decade. For example, the following journals and conference proceedings were included: IEEE Transactions on

Parallel and Distributed Systems, Computer Networks, Wireless Networks, IEEE/ACM Transactions on Networking, and Ad Hoc Networks; the IEEE Conference on Computer Communications (INFOCOM), the International Conference on Distributed Computing Systems (ICDCS). We observed that some of the techniques have multiple publications that were considered as improvements to the same technique, for example, TinyDDS has three papers titled “Middleware Support for Pluggable Non-functional Properties in Wireless Sensor Networks” [29] , “Self-Configuring Publish/Subscribe Middleware for Wireless Sensor Networks” [30], and “TinyDDS: An Interoperable and Configurable Publish/Subscribe Middleware for Wireless Sensor Networks” [21]. In such cases, we counted them as one technique and the enhancements were taken in consideration in the comparison. During the searching process, we found that some of the papers do not propose a new technique of pub/sub in sensor networks, rather they describe some issues related to the subject, for example, data matching algorithms were thoroughly investigated by Heidemann et. al. [31]; these papers also included in the description part.

2.1 Pub/Sub Model Overview

In this section we review the pub/sub interaction scheme in the context of sensor networks. The main components of any pub/sub system are described and their implementation challenges in sensor networks are highlighted. Also, pub/sub model variants are identified in terms of subscription model and notification service architecture. We used the information presented here in the classification of the existing solutions of pub/sub scheme in WSN/WSAN applications in the next section.

2.1.1 Principles of The Pub/Sub Model

The pub/sub interaction scheme is proposed for large-scale distributed systems to make them flexible, scalable, and faster. Figure 4 depicts the basic model of pub/sub system and its main components. The core component is the pub/sub service or notification service that mainly provides storage service and manages the subscriptions. As illustrated in Figure 4, data logically appears as a global data space whereas in real implementations it is distributed over the system end-nodes and/or brokers, i.e. centralized servers. The notification service acts as a mediator between publishers (producers) and subscribers (consumers). The subscriber who is interested in a certain event, for example in Figure 4 like E1, E2, or E3, has the ability to express his interest by using subscribing function $\text{sub}(E)$, and subsequently the notification service matches the subscription with the existing events which have been published by the publishers, and delivers the matched event to the subscriber. Three main operations are used in publish/subscribe systems: (1) $\text{pub}(E)$ to publish the events, (2) $\text{sub}(E)$ to subscribe to a certain event, and (3) $\text{unsub}(E)$ to unsubscribe to an event. The participants could be either a publisher or subscriber or both at the same time as depicted in the Figure by pub/sub entity. Eugster et al. [7] described the pub/sub model in more detail.

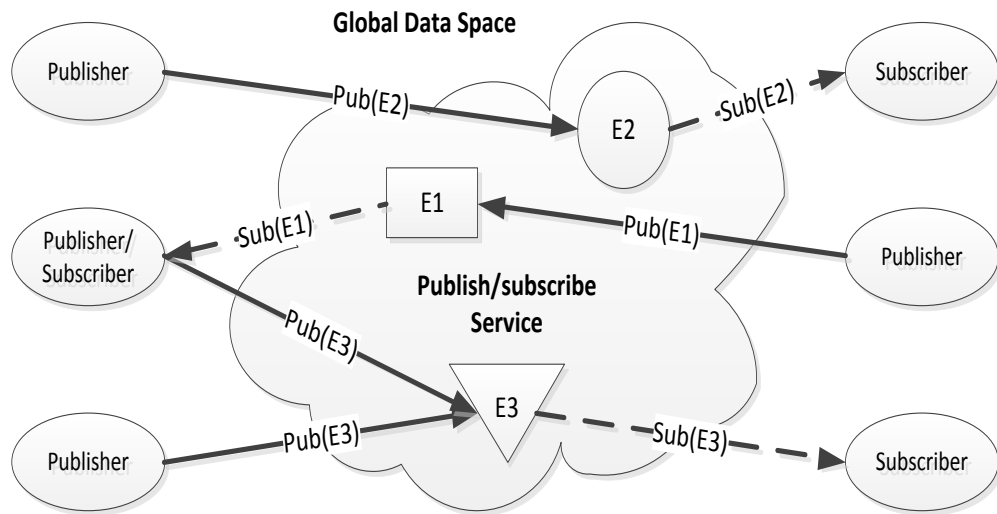


Figure 4. Simple architecture for publish/subscribe communication model

For flexibility and scalability, pub/sub service provides the decoupling between publishers and subscribers in three dimensions named as decoupling properties [32]:

- *Space dimension*: the interacting entities (publishers and subscribers) do not need to know each other because their main interest is in the event itself no matter from where it comes. The pub/sub service is the mediator between publisher and subscriber; where the publisher publishes events through the pub/sub service and the subscriber gets the events indirectly from the pub/sub service.
- *Time dimension*: the interacting entities do not need to be actively participating in the interaction at the same time. For example, the publisher can publish certain event while the subscriber to that event might come after a while or even after the publisher life time is over; also the subscriber might subscribe to certain event that has not been published yet. That is very useful for high dynamic networks such as high error-prone wireless networks where the nodes disconnection rate is high.
- *Synchronization dimension*: that means no blocking in both sides (publisher and

subscriber); while executing some concurrent tasks, publishers and subscribers are not blocked during publishing or subscribing to events. In contrast, in synchronous communication paradigms the end node is blocked until the other node receives the message, which leads to rigid and static applications.

Distributed systems are asynchronous by nature, such as mobile systems [33] and sensing dynamic environments in WSN [34]. Removing dependencies between the interacting participants makes the decoupling properties significantly increase the scalability of these systems and make them faster.

2.1.2 Pub/Sub Middleware Components

In this section we describe the main components of pub/sub system within the context of sensor-based networks. As shown in Figure 5, the Pub/Sub system consists mainly of five components: 1) the programming abstractions and APIs (Application Programming Interfaces), 2) end nodes which are publishers and subscribers, 3) event/query (publications/subscriptions) messages, 4) pub/sub service (Notification service), and 5) QoS mechanisms that could be supported by pub/sub applications.

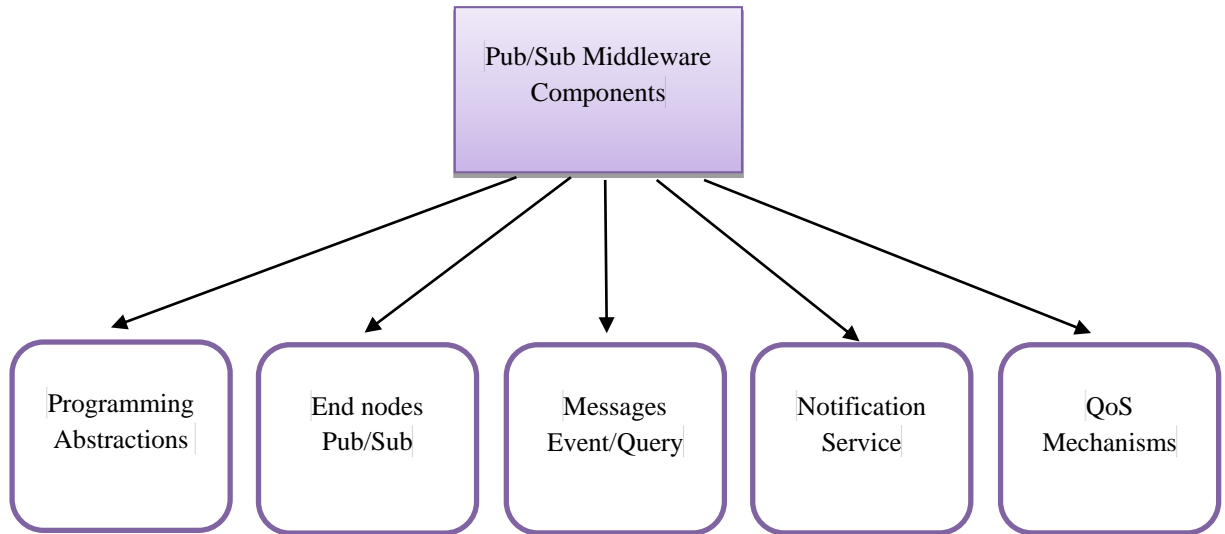


Figure 5. Main components of publish/subscribe middleware

Programming abstractions

In order to reduce the complexity and increase the efficiency of the WSN applications development, programming abstractions are introduced in the form of Application Programming Interfaces (APIs). For example, in Pub/Sub based applications, which are mostly event-driven applications [35], the main middleware APIs (Application Programming Interface) are provided to create, publish, subscribe, and unsubscribe a certain event. These programming abstractions are built to ease the application development by hiding the heterogeneity and the detailed and complex information of the underlying network layers from the application developers. For example, a sense-and-react application program can be developed easily by writing six instructions in Maté middleware [36]. There are two main abstraction levels regarding WSN applications, the node level and system level. At the node level, the developer has a fine-grained control on the network, where he can program the action and cooperation of the individual SA (Sensor/Actuator) devices [37]. Thus, this level of abstraction supports the developer to build more efficient WSN applications in terms of resource allocation and power consumption. At the system level, the WSN abstracts to one

single virtual system with global behavior, which makes the task easier but with less control to the Sensor/Actuator (SA) devices [38] [39]. A centralized program is built, where the developers concentrate more on the whole system functionality without bothering themselves with the SA devices coordination mechanism.

End-nodes

Any communication system has two end users, the sender and the receiver; in Pub/Sub system we call them the publisher (sender) and the subscriber (receiver). The publisher creates the events and sends them to the notification service which in turn sends it to the interested subscriber. If there is no interested subscriber, the event is stored in the notification service by means of events table until either a new subscription is received or it reaches its expiry data. The subscriber creates the subscriptions and sends them to the notification service where matching process is triggered to search for a matching event. If no matching event is found, the subscription is stored in the subscriptions table until a matching event is found or it reaches its expiry date. In WSAN, see Figure 6, the system consists of four main entities (publishers and subscribers): sensor, actuator, sink, and the application (end user). These entities are distributed over 3 virtual layers; each layer has different hardware and software capabilities. As a result, different versions of Pub/Sub middleware are distributed over the 3 layers. If we take the SA device main function into consideration, we would consider the sensor as a publisher and the actuator as a subscriber. However, in fact, all four entities are publishers and subscribers at the same time. For example, sensor nodes publish collected data and subscribe to control signals, e.g. sleep or wakeup signals, and also to software updates.

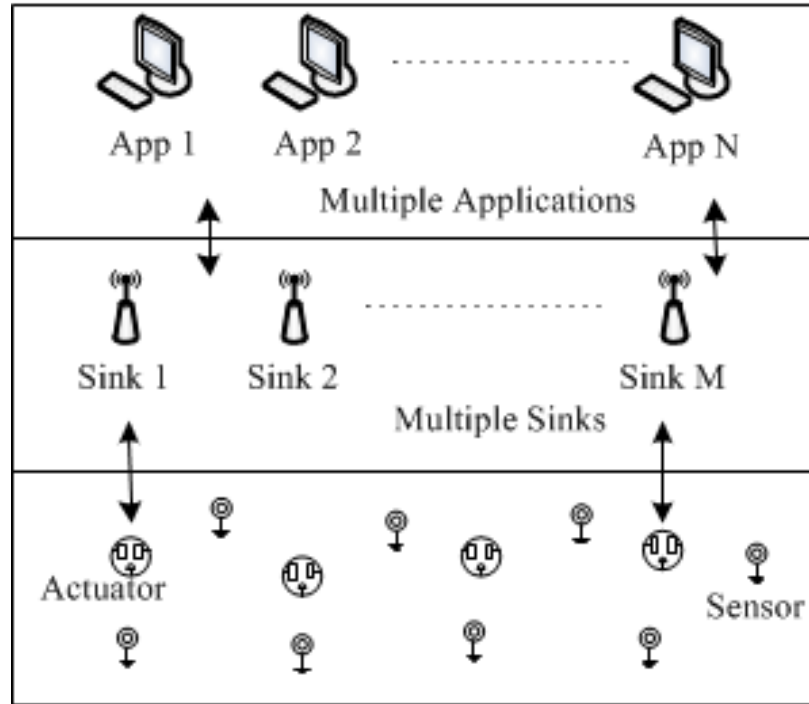


Figure 6. The 3-Layers WSN architecture

Messages (Event/Query)

There are three main message types in the pub/sub interaction paradigm: advertise, event (publication), and query (subscription). Advertise messages are used to advertise the events before the publication, such as in Mires [8] and MQTT-S [11]. These messages, created by the application, include message header and payload (user-data) message, and typically have main fields in the message header such as identifier, issuer, and some fields related to the QoS parameters supported by the application such as priority, deadline, and expiration time. The message format varies from one implementation to another; for example, some solutions represent the message as an array of bytes like in IBM MQSeries [40], or use a set of types, e.g. text or XML, as in DDS [41] and MQTT-S [42], or allow the programmer to create his/her own message structure, e.g. TEBCOO [43]. Figure 7 illustrates the general message format and gives the average size of the packet header in pub/sub solutions in WSN such as in TinyDDS [21], Mires, and PSQUASAR [19].



Figure 7. General message format

The query (subscription) message is very important since the way it is expressed can be used to classify the most widely used Pub/Sub systems. At the node level, a higher level of expressiveness requires more computation power and advanced algorithm designs. However, at the network level a higher degree of expressiveness leads to higher performance due to the reduction in the consumed bandwidth by eliminating the unwanted information. Usually subscribers receive the events which they are interested in; they do not register to all events but to some or to patterns of them. From the event expressiveness point of view, the ways the subscribers express their interest in those events differ from one implementation to another and directly affect the architecture and the algorithms used for notification service implementation. Figure 8 shows the four common schemes of expressing events in the Pub/Sub interaction model, which are: channel-based, topic-based, content-based, and type-based. In this study, we call these Pub/Sub systems since we will use this classification to distinguish between the surveyed WSN Pub/Sub protocols. However, their name varies from one study to another, e.g. Pub/Sub variants [7], subscription models [24], and event subscription [32].

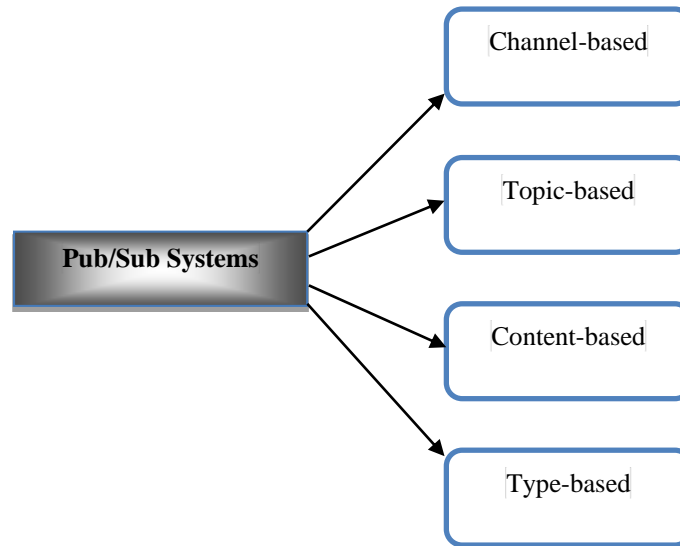


Figure 8. Pub/Sub schemes

Channel based. A channel-based system groups the events (notifications) under different channels, such that the subscribers only have to subscribe to the channel that includes the events they are interested in. The main difference between this approach and topic-based is that no topic name is associated with the published events; instead a channel-id is embedded to each published event. Thus, publishing an event to a specific channel implies broadcasting this event to all the subscribers who have subscribed to that channel, and vice versa. Java Message Service (JMS) [44] is a concrete example for such approach. In JMS, a queue structure is used to implement channels in the notification service broker (a centralized server). For example, if the publisher publishes an event with a specific channel-id (queue-id), the broker searches for the queue-id associated with the event and insert it to the queue on a FIFO (First In First Out) basis. On the other side, the subscriber subscribes to the channel by specifying the queue-id (e.g., `queue_name==queue-id`); then the broker will immediately route the events that just come from the publisher to the subscriber who has the same queue-id. When the event has been received by the broker, it will check if there is no queue with the same event queue-id then it creates a new one. Other examples of industrial implementations

are the CORBA notification service [45] and CORBA event service [46] [47]. To the best of our knowledge, no channel-based Pub/Sub solution for WSN/WSAN has been proposed in the literature. That might be because the WSAN applications are resource-constrained and tend more to the fine-grained schemes like content-based subscriptions. That is because a content-based system provides a fine-grained control on the event contents, which significantly reduces the overall network traffic and consequently minimizes the bandwidth and energy consumption.

Topic based. The topic-based system extends the notion of channel-based scheme by adding more classification and characterization for the event content [7] [48]. The topic-name corresponds to channel-id, where it forms a logical channel that connects the publisher to all subscribers who are interested in this particular topic. A fixed set of topics are made in the development stage (static subscription model), and the publisher tags the notification with a unique topic-id which is used by the notification service in the matching process to get the corresponding subscriber who is interested in the published topic. To make this scheme more expressive, a hierarchical approach is used to go further in event content classification [49] [50]. In this way, the topic can be further divided into sub-topics using a tree structure. For example, topic A can be divided into sub-topics B and C, thus topic A is the root node in the tree and has two children B and C. Thereby, during matching process, all the events that match B will be sent to all subscribers of topic A and sub-topic B. A concrete example for topic-based scheme is the OMG DDS standard [41], where topics could be implemented using C++ struct type that includes the topic name of type string, as in RTI Connext product [51] [52]. Each individual topic has a unique keyword and each topic includes multiple instances where each instance inherits the topic attributes and identified by a topic key

attribute which can be any field within the topic, for the example in Figure 9 the keyword can be the color. Other implementations from industry are iBus [53], and TIBCO Rendezvous [43]. Many research proposals have been introduced in the literature on the topic-based Pub/Sub middleware in WSN; for example, Mires [8], PS-QUASAR [19], and TinyDDS [21]. The advantage of this approach is the potential to simply use the existing group-based multicast techniques, e.g. IP Multicasting [54] [55] or an equivalent overlay multitasking facility [56] [57], by assigning every topic to a multicast group. For more information about pub/sub multicast techniques see [58] [59] [60].

```
struct Shape{  
    string<128> color; //@key  
    long x;  
    long y;  
    long shapsize;  
};
```

Figure 9. Topics represented by C++ structures

Content based. The content-based system is a fine-grained control approach that increases the expressiveness degree of the subscriptions. The main difference between the topic-based and content-based is that the subscriber can express its interest in a more dynamic and accurate way, where the topic-based (even the hierarchical) approach offers static and limited expressiveness. Also, in topic-based scheme the content of the published event is hidden to Pub/Sub service except for the topic-id, whereas in content-based scheme, it is aware of the published event contents (attributes). As a result, the subscriber can filter out the topics that it is not interested in by putting conditions (constraints) over the content (the values of the topic attributes) of the subscribed topic.

To illustrate the difference between topic and content based schemes, we describe a practical example in heat monitoring systems. In this system, the sensors (publishers) are deployed and periodically read the data (Temperature) from different places (e.g., machines or rooms), and the actuators (subscribers) are distributed over the environment to do specific tasks such as controlling the alarm and cooling system. In our example, see Figure 10, we have multiple sensors (S_1, S_2, \dots, S_n), two actuators (A_1 and A_2), and one sink node that is attached to historical data base server, which is used to store the sensor readings and distribute them to the end users' monitors for further online monitoring and data analysis. In content-based systems, it is allowed to subscribe to an event with applying particular constraints using comparison operators (e.g., $=, <, >, \geq, \leq$). The sensors publish a topic m (e.g., particular machine temperature), and each subscriber (actuators and sink) receives different patterns from the published topic based on their predefined interest, as in the following example:

- The sink node receives the topic m as is without applying any type of filtering.
- The alarm actuator receives the topic m when the temperature degree is greater than some threshold (30°)
- The cooling valve receives the topic m when the temperature degree is greater than (50°)

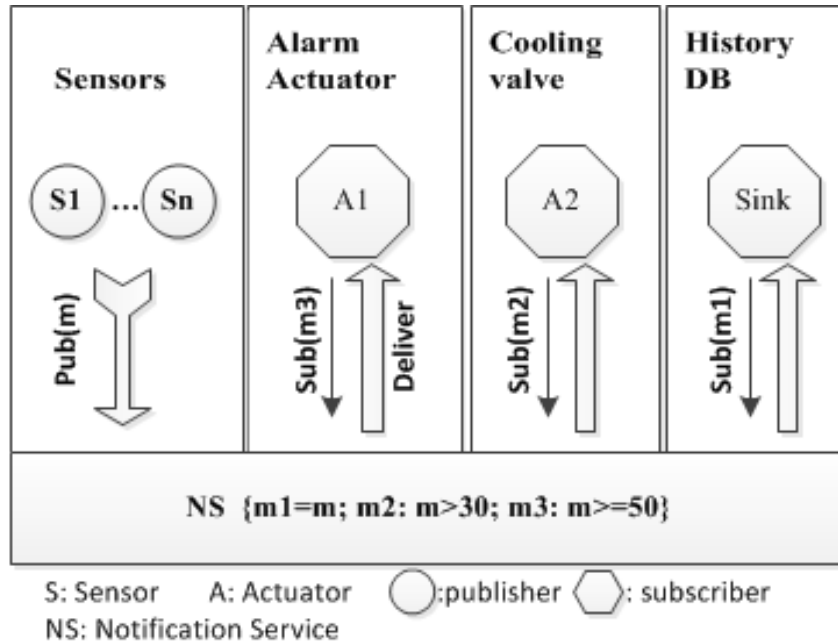


Figure 10. Content-based vs. Topic-based Pub/Sub interaction

There is a tradeoff between the high performance behavior in terms of delay and resource consumption and the degree of expressiveness, and the design and implementation of such filtering algorithms is not that easy. A lot of algorithms in Pub/Sub infrastructures have been proposed to minimize the overhead and time consumed by the content filtering process [61] [62] [63] [60] [64] [65] [66]. In resource-limited systems like WSN, the event filtering process significantly affects their performance, specifically for real-time applications. On one hand, it increases the processing overhead (simple and fast algorithms is better) and adds more end-to-end delay. On the other hand, it reduces the total bandwidth consumption which, as a result, increases the network performance in terms of delay and throughput. Several content-based protocols for WSN have been proposed, for examples, the MQTT-S [11], TinyDDS [21], μ DDS [67], Dv/DRP [68] and TinyMQ [69]. For applications where the event space can be divided to limited set of possible discrete values, it is better to use the topic-based scheme to avoid the additional overhead caused by content filtering mechanism.

Type based. Another event subscription model is presented by Eugester [70] [71] named as type-based scheme. Instead of subscribing to topic name (e.g. topic == “temperature”), in typed-based the subscribers subscribe to the events that have a particular structure or type. For example, in Figure 9 instead of subscribing to the topic (color name), the subscriber will subscribe to the structure name, which is shape in this example. Thus, the subscriber will receive all the events that have the same structure illustrated in Figure 9.

Notification service

In Pub/Sub systems, the responsibility of data dissemination lies on the Notification Service (NS) component. It is the heart of Pub/Sub middleware, where it mediates and coordinates between publishers and subscribers. It interacts with the publishers and subscribers through specific operations as illustrated in Figure 11. The publisher uses `publish ()` and `advertise ()` for publishing and advertising new topics; and the subscriber uses `subscribe ()` and `unsubscribe ()` to subscribe and unsubscribe to a particular topic, and the NS uses `notify ()` to notify the subscriber with the matched topic. The main services include storing the publications and subscriptions, managing Pub/Sub Quality of Services (QoS), discovering the participants (publishers and subscribers), filtering the events based on the subscriptions constraints, matching the subscriptions with the publications, and routing the events based on the matching results, see Figure 11 . Each of these services is still an open issue for research especially for limited-resources systems such as WSN. According to Carzaniga et al. [61] the two main services are (a) the matching service where it determines which publications match with subscriptions, and maintain that with matching tables; (b) the routing service where it routes the matching publications from the publishers to the relevant subscribers.

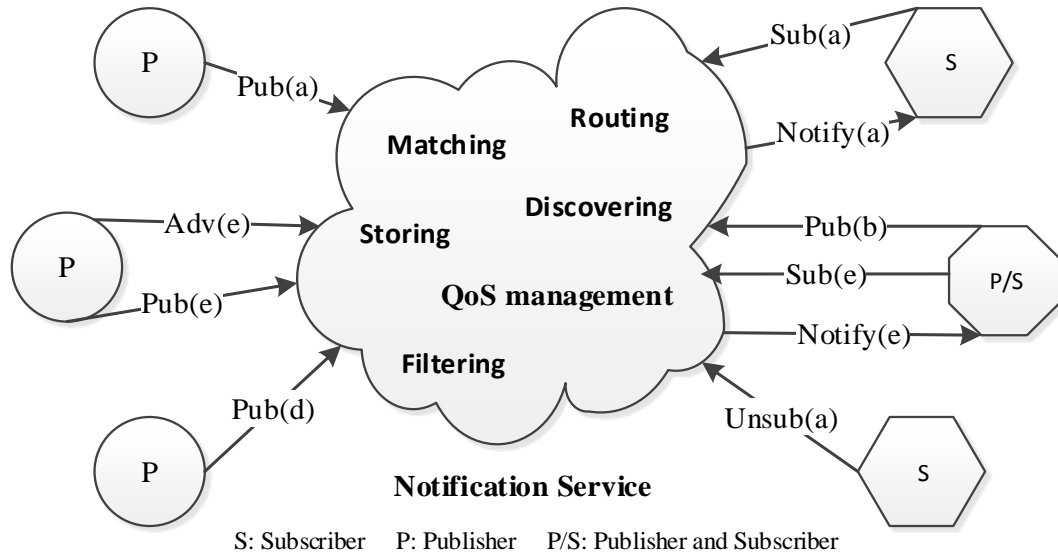


Figure 11. Notification Service

Matching. The matching service is the process of checking the published events against the subscriptions to decide whether to send the event to the subscriber or not. In topic-based systems, this process has small effect on the overall performance, since the comparisons just including the topic name without going deeper to the event specific attributes (event content). On the other hand, it causes significant performance degradation in case of content-based systems in terms of delay and resource consumption (e.g., CPU cycles). The number of matching cycles increases exponentially with the number of subscriptions and the maximum length of the matching cycle, which needs a large amount of space to store intermediate results [72]. Several studies have been conducted to mitigate this problem, and propose efficient matching approaches especially in the content-based Pub/Sub applications [73] [74]. Rajibi et al. [75] have classified the matching algorithms to two main categories: predicate indexing and testing network. The predicate indexing algorithms [63] [76] [77] [62] consist of two phases. The first phase determines all the predicates (in all subscriptions) that are satisfied by the event. The second phase finds all the subscriptions that are matched

by the event based on the results of the first phase. The algorithms based on testing networks pre-process the set of subscriptions into a matching tree. Events enter the tree at the root node and are filtered through by intermediate nodes. An event that passes all intermediate testing nodes reaches a leaf node where a reference to a matching subscription is stored [78] [79]. One recent research [65] has exploited High Performance Computing (HPC) technology and propose a parallel algorithm to avoid the drawbacks of the traditional sequential search. In [31], the authors classify the matching point based on the application, if the publishers are less than subscribers the matching is better to be at the publisher side and vice versa. Although a lot of research has been done on matching techniques, to the best of our knowledge, no study has tried yet to examine those algorithms on WSN infrastructures; and no one from the WSN Pub/Sub solutions that we have surveyed had mentioned the matching algorithms used, except TinyCOPS [80] . Therefore, one of the open research directions is to investigate the suitability of such algorithms in resource-constrained applications, and to find the best matching point based on the nature of the application.

Routing. The proposed routing protocols for pub/sub sensor networks can be categorized into three main categories, flooding, selective and gossiping routing techniques. In flooding routing, either the publisher or the subscriber will broadcast its publications/subscriptions to the whole network. Unlike the fully deterministic flooding routing, the gossiping-based routing is fully probabilistic, random approach where a random neighbor is selected for sending the packet to, and then this neighbor will randomly select one of its neighbors and so on. In between, the selective routing combines both the broadcasting and random walk techniques, e.g. the semi-probabilistic approach proposed by Costa et al [81], or the semi-broadcast approach that reduces the subscriptions propagation in response to broadcast part

of the publication content [82].

The flooding routing protocols are simple to implement and do not require any state information, e.g. routing tables, to be saved in memory; however, it overwhelms the network by message overhead which in turn increases the collisions in the network. Therefore, this type is more suitable for applications when a large number of subscribers are interested in most of the events, and when the subscriptions change at low rate [61]. In gossiping, broadcast overhead of flooding approaches is mitigated by avoiding broadcast transmissions and use random walks to reach the destination. Thereby reducing the traffic overhead of the flooding approaches in the cost of adding time delay and probabilistic guarantee to reach the destination. The selective routing protocols combines the advantages of both types by using broadcast to some predefined extent and random path selection. A data-centric energy aware routing in WSN is also proposed in the literature [83], which depends on the residual energy for routing process. For further information about these routing classes the reader is recommended to refer to the survey studies in [24], [84] [63] [85] [58] [86].

QoS Mechanisms

The communication medium provides guarantees to support qualities of services, where these guarantees vary strongly between different systems [7]. One of the advanced features of any WSN middleware is support for Quality of Service (QoS). Unlike the direct connection between sender and receiver, in pub/sub model the decoupling properties make the system behavior less deterministic. As a result, providing QoS support in pub/sub systems is not an easy task [87]. Moreover, providing QoS support in resource-constrained networks is even more challenging issue, where in sensor networks it is still an open issue for research

[88] [26]. QoS can be expressed in the application layer by data accuracy, aggregation delay, coverage, and system lifetime; whereas in the network layer by latency, throughput, bandwidth utilization, message delay, jitter, and loss. If the QoS requirements in the application layer cannot be satisfied by the network layer the middleware should negotiate between the two layers to get a new QoS guarantee [89]. Recommended references on QoS support in WSN are [20] [90] [88]. Among of the most common QoSs provided by pub/sub model, we selected the most relevant for WSN and those used in previous works. These QoSs are as follows:

Reliability. specifies the ability of the network to ensure reliable data transmission between nodes. Information in pub/sub application needs to be transmitted in a reliable way to make sure that important measurements, alarms, or notifications generated by the system reach their desired destination.

Priority. defines a way to assign different level of importance to the data flows. In this way, the more important the data is, the sooner the system will try to process it. In WSN systems, usually different levels of importance are associated with the messages exchanged between nodes. For example, monitoring readings does not usually have the same importance as failure or attack notification events.

Deadline. is also known as maximum allowed latency. It defines a maximum length of time the subscriber will wait for an update. In real-time systems, if data received beyond a certain threshold, it would not make sense, and thus will be dropped. This is used in event transmission scheduling in which Earliest Deadline First (EDF) algorithm can be used.

Energy-awareness. WSN devices rely on battery energy which are limited and in most cases batteries are very difficult to change. The energy is mostly consumed in the wireless transmission, since the energy consumed by sensing and computation is very little compared to transmission. Therefore, the transmissions have to be managed sensibly to minimize the energy consumption in order to maximize the network life time. Consequently, handling the duty cycles of the SA devices is a critical issue. The WSN devices need to go to sleep mode or even deep sleep mode whenever they do not have new data, and then wake up and publish whenever the new data arrives. In most applications, the sleeping time could potentially be a very long, ranging from several seconds to hours. The middleware techniques should be aware of this to save energy as much as possible. However, energy efficiency and QoS support are two conflicting requirements and the WSN design needs to efficiently set the tradeoffs between them [90]. For more information the reader should refer to previous works on energy efficiency techniques in WSN, e.g. [91] [92].

2.2 Pub/Sub in WSN

In this section we discuss the pub/sub based solutions for WSN/WSAN in the past years. We focused on gathering the most important information about each technique including its main features, components, architecture, and drawbacks. A comparative study was then conducted to show the pros, cons, differences, and similarities of those techniques. The pub/sub WSN general reference model is presented at the end of this section.

2.2.1 Existing Solutions

Directed Diffusion [93] is the earliest pub/sub communication paradigm for wireless sensor networking. It is a data-centric protocol in that all the communications concern named data

that is described by attribute-value pairs. As any pub/sub system, it has almost the same common elements and functions, Figure 12 depicts a simplified scheme for this paradigm. The subscriptions are called interests, and are broadcasted throughout the whole network. During the subscriptions dissemination, gradients are set up within the network, to be used later to draw the events (requested data). Each node examines the interest and do a matching process locally. If it has the requested data, then it sends back the information to the sink by reinforcing the reverse path of the interest. Otherwise, it just propagates the interest through the network. Thus, the matching process is distributed and does not need a centralized broker. This avoids the disadvantages of the centralized processing (which is not suitable for sensor networks) and evenly distributes the energy consumption. However, it will add 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; it is a direction state that is directed towards the node sending the interest. Recently, a secured version of this protocol has been proposed, it provides authenticity and integrity with a relatively low overhead [94].

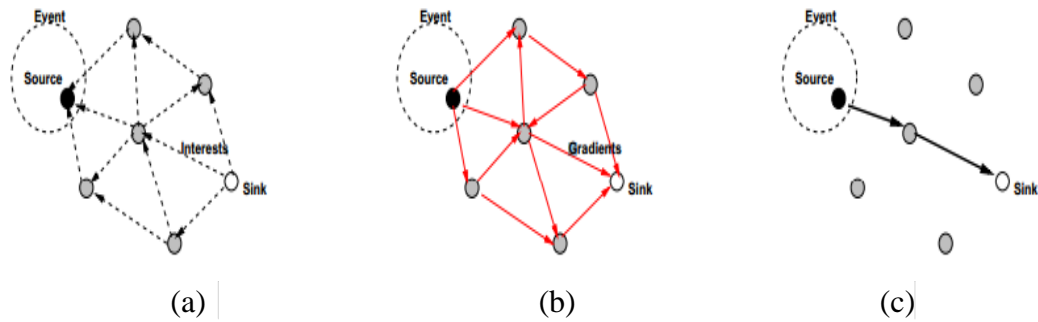


Figure 12. Directed Diffusion simplified schematic. (a) Interests propagation. (b) Gradients setup. (c) Reinforced path.

Mires [8] is a pub/sub middleware for WSN. *Mires* is designed mainly to facilitate the development of WSN applications. It was implemented on top of TinyOS [95], an event-based operating system for sensor networks. In *Mires*, each sensor advertises its available topics (e.g. temperature, pressure, luminosity or humidity) to the user applications through the sink node. Hereafter, each application selects the topics that it is interested in and broadcasts the subscription into the network. The sensors then match their topics against the subscriptions and send the matched data to the interested application. An aggregation service is provided to minimize the overhead of the transmitted messages. Although, a Multi-Hop routing protocol is included with the middleware, any multi-hop routing protocol can be added as long as it implements the required interface by *Mires*. As illustrated in Figure 13, the architecture is fully distributed over the network nodes (no centralization). However, it has some limitations; it is made for traditional WSN where a single sink controls the network behavior and collects the data from sensors to end-user applications; also it does not support actuator, QoS, or energy-aware mechanisms. Moreover, no performance evaluation has been published for this solution.

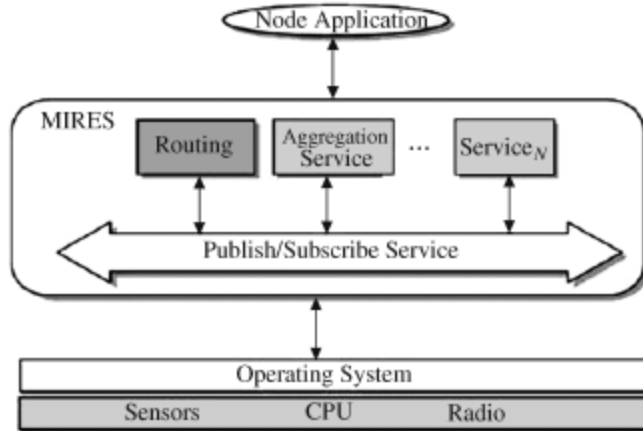


Figure 13. Mires Architecture

MQTT-S [11] is an IBM pub/sub protocol that was invented by Stanford-Clark and Hunkeler in 1999; it is named as MQTT which stands for Message Queuing Telemetry Transport [96]. Telemetry means remote data transmission to monitor environmental conditions or equipment parameters. It is an extremely simple and lightweight messaging protocol designed for constrained devices and low-bandwidth, high-latency or unreliable networks. Consequently and due to its lightweight properties, an extension version of MQTT protocol was proposed for wireless sensor networks [97]. The main goal was to simplify the integration of the WASN with the enterprise networks by extending the enterprise pub/sub middleware protocols into the WSN infrastructure. The pub/sub service (notification service) is located in brokers that use the original MQTT protocol, where the SA devices software is kept as simple as possible; Figure 14 illustrates the MQTT-S architecture. The SA devices use the collection tree protocol (CTP) [98] as its underlying routing protocol which allow any device to send data to the closest gateway. Reliability QoS is implemented at three levels: (1) best effort (send just once either successfully received or not), (2) retransmit until the message is acknowledged (may incurs redundancy), (3) assure no redundancy. Several

drawbacks exist in this solution; for example, the broker architecture raises the centralized approaches problems such as single point of failure and bottleneck. Also, the translation from gateways (MQTT-S) to broker (MQTT) incurs more delay which increases the potential of considering this protocol being unsuitable for real time systems. Moreover, the protocol does not support or evaluate sleeping modes for energy saving purposes.

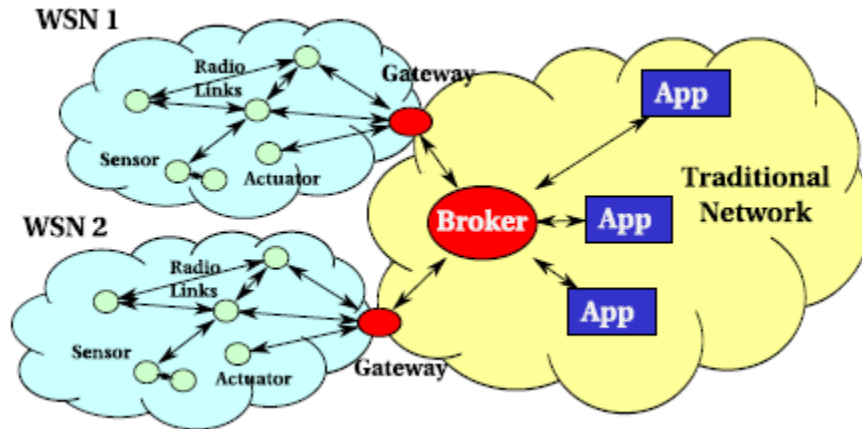


Figure 14. MQTT-S architecture

TinyCOPS [80] 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: (1) communication protocol components for subscription and notification delivery, (2) supported data attributes, and (3) set of service extension components. As directed diffusion, it uses an attribute-based naming scheme; this scheme is augmented with metadata information that is provided through pub/sub API and is used to send control information to the publisher, e.g. sensing rate, and to add additional communication control information (timestamps, message sequence number, etc.) for the service extension components. The service extension components (SEC) 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.

Figure 15 depicts the high-level decomposition of the framework.

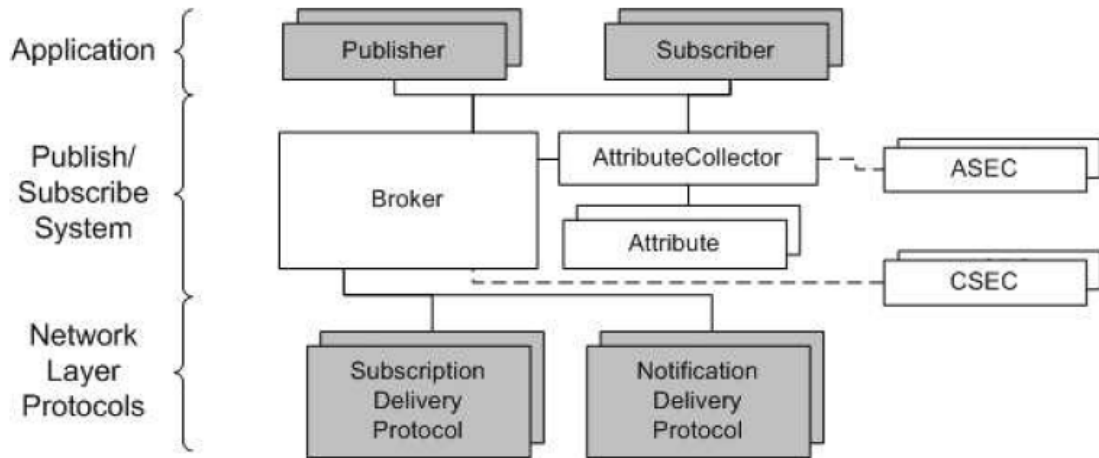


Figure 15. TinyCOPS architecture

PS-QUASAR [19] is a pub/sub middleware that focuses on providing QoS (reliability, deadline, priority) support and high level programming model to the WSN 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 consists of three different modules: maintenance protocol, routing module, and API. Figure 16 depicts the *PS-QUASAR* architecture and shows how the three modules are inter-connected. The maintenance protocol is in charge of creating the links between neighbor nodes, and discovering pub/sub end nodes (publishers and subscribers). The information collected from the maintenance protocol is used by the routing module to route the events. A topic-based (less matching overhead than content-based) programming model API is used to provide a set of methods for developers to develop WSN applications using *PS-QUASAR* middleware. The Bellman-Ford

algorithm [99] is enhanced and used to build a routing tree protocol where each node maintains a routing table. Although PS-QUASAR provides QoS-aware, energy efficient, and robust protocol, the cost of such mechanisms would be in memory space, a very critical resource in SA devices. Thus, memory footprint was one of the most important performance evaluation measurements that the paper should have considered. Also, performance evaluation considered only deterministic behavior in topology (deployments) and data rates, while most of the WSN applications require random distribution for sensor nodes.

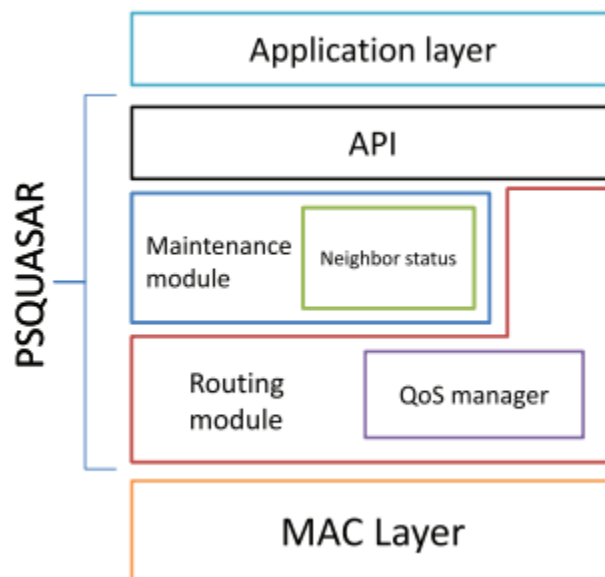


Figure 16. PS-QUASAR architecture

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 [18]. The system main components are illustrated in Figure 17; where it is composed of stationary (sensors and traditional network) and mobile networks (mobile phones). The internet users can access the WSN data anytime from anywhere (Ubiquitous) through platform server (broker). The sensors are distributed over the monitored area and publish the

data to the mobile phones which then send it to the interested subscribers (internet clients) via mobile phone networks, e.g. 3G. The proposed solution was tested using an outdoor test-bed, and a hiking trial monitoring application was developed on top of the PSWSN-MM middleware. The application can provide the subscribers with sensing data such as temperature, humidity, light intensity, and hiking speed; such that they can decide whether to go for hiking in that particular area or not. Due to its reliability mechanism, where a packet is not sent until the previous one is acknowledged, the system is not suitable for the real time systems. Moreover, the system lacks other QoS mechanisms support like priority and deadline.

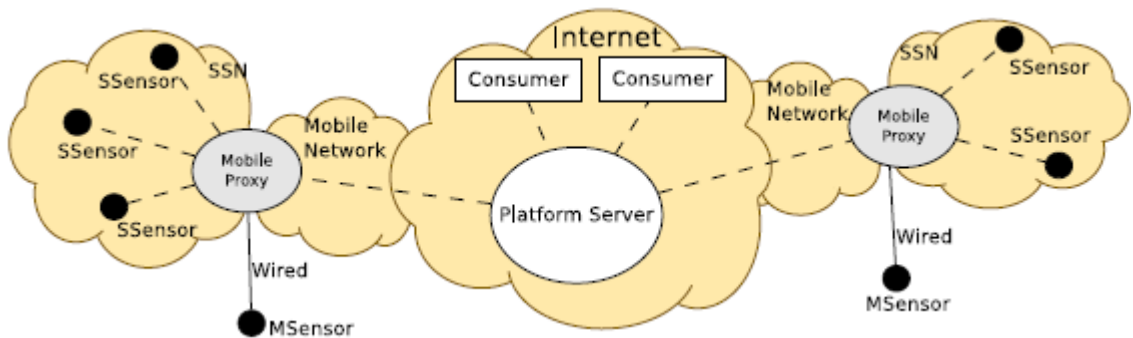


Figure 17. UPSWSN-MM publish/subscribe system

PUB-2-SUB+. Unlike the gossiping routing where message content is ignored during the dissemination, Pub-2-Sub+ [100] is based on content-guided routing to offer better efficiency (less storage and communication costs) than the traditional approach. It is based on a naming scheme [101] designed for content-based pub/sub for WSN. Pub-2-Sub+ maintains a set of m spanning trees, each rooted at a node in the network. The root nodes are dedicated reliable nodes placed at random network positions. Each tree corresponds to a naming tree assigning a binary-string name to each node; hence, a node has m names. The names on a tree form a prefix tree. Based on the naming scheme, each node is assigned a “zone” of binary strings to

own. The zone of a node is the set of all binary strings starting with this node's name but not with any child node's name. A query is subscribed to a random tree and an event is published to all the trees. Pub-2-Sub+ formats an event as a binary string (e.g., '0110010') and a query as an interval of binary strings (e.g., ['0110001', '0110101']). On the randomly chosen tree, a query is routed to, and stored at, all the nodes whose zone overlaps with the query's interval. On each tree, an event is published to the node whose name is the longest prefix of the event string. In general, the notification path is bounded by two times the tree height which should be $O(\log n)$ in most cases. Also, because there are multiple paths for event notification, the disconnection of a path due to some failure does not stop an event from finding its way to the matching queries.

TinyMQ [69] is a content-based pub/sub middleware for WSN. It is considered as an improvement for the PUB-2-SUB+ solution by adding content-based routing and avoiding the congestion at the sink (tree root) by using multiple sinks. An overlay structured network is constructed to route events/publications and queries/subscriptions without location information. The network is logically connected by assigning virtual addresses (unique keys) to all network nodes and using naming scheme based on binary strings. The unique keys represented by the binary strings chosen from $\{0, 1\}$ are used as the logical addresses to enable hash based content-based message matching and routing. This matching approach guarantees that the events meet the queries in the certain rendezvous nodes. First the network is divided into m-tree based clusters and each cluster contains one tree with sink node as a root, and there is no overlapping between the clusters (trees), the trees are constructed using a maintenance protocol.

The system architecture consists of two layers:

- 1 Overlay network layer: a logical topology is constructed despite of the network churn or node failures. A naming structure is used to assign a virtual unique address to each node.
- 2 Pub/Sub layer: Provides message mapping (subscriptions with publications) and message routing (subscriptions, publications, and notifications) services, where routing is based on the virtual addresses of the nodes.

TinyMQ provides interoperability among the nodes in WSN (not with the traditional networks, e.g. enterprise network). Also, similar to gossiping protocol [102] it provides none location-based information dissemination, and better in the sense of content-based routing, where gossiping routing ignores the message content. However, TinyMQ does not support actuators and QoS to WSN. Furthermore, the cost of the algorithm in terms of energy consumption and communication overhead was not evaluated, although it is an important performance measurement in such limited resources systems.

TinyDDS [21] 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, it allows WSN applications to have fine-grained control over application-level and middleware-level non-functional properties and flexibly specialize in their own requirements. 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 great services and support for WSN, a complete and robust DDS system for WSN is yet to be developed [20]. TinyDDS lacks energy saving mechanisms and energy consumption evaluation because it is still not lightweight enough to fit the WSN requirements. The TinyDDS architecture and main components is depicted in Figure 18.

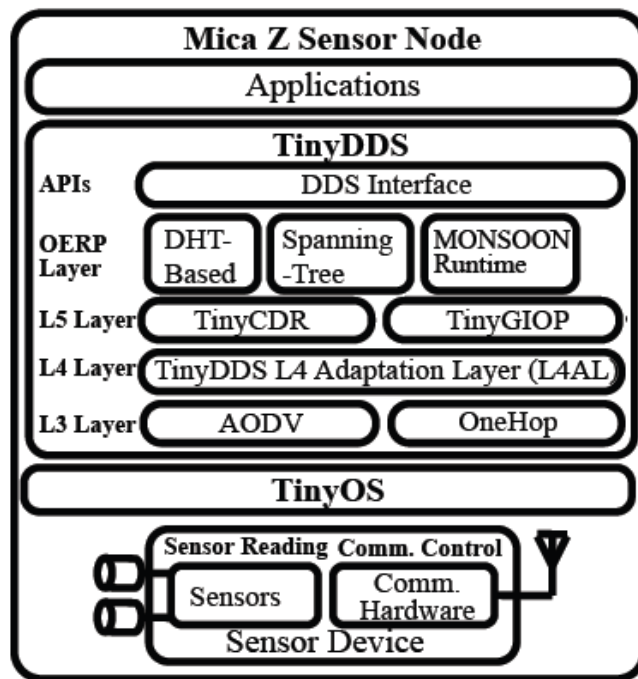


Figure 18. TinyDDS architecture over TinyOS and MicaZ platform

Quad-PubSub [17] is a pub/sub solution for WSN that exploits the location-based addressing scheme to offer support for the transparent operation of resource-aware routing. It aims to minimize the communication costs by targeting the shared event dissemination paths, and balances the routing load over multiple paths to overcome energy hole problem and, thus, increasing the network life time. Quad-PubSub uses localized resolving algorithm that is easy in operation and comprising distance calculations. This algorithm iteratively resolves the

sub/unsub operations over the network. It establishes paths without the involvement of end-point publishers or subscribers, where it decouples the publishers and subscribers using a set of intermediate nodes, called Event Brokers (EB). The network area is divided into sub-areas each of which is controlled by EB. The subscriptions forwarded to the EB that matches them with the published events in its area and serve the interested subscribers with the matched events. Thus, the data dissemination is distributed among the EB to balance the communication load. However, this may make the EBs die first before the other nodes and significantly affects the network connectivity. Although the protocol aim is to reduce the energy consumption, there is no evaluation in the paper for energy consumption. Moreover, no QoSs are supported and the implementation is highly abstracted.

The comparison of the reviewed solutions is summarized in two tables. Table 1 and Table 2 compares the proposed prototypes based on the criteria that were discussed in the pub/sub model overview section; and summarizes the implementation and evaluation issues of each proposed solution. One of the most important issues that should be extracted from the proposed solutions is the methodologies used to verify and evaluate their performance. Several surveys have been done in WSN simulators that are used in the literature such as [103] [104] [105]. To make this part fully self-contained, we present the mostly used simulators in the literature for evaluating pub/sub solutions of WSN/WSAN. Table 3 summarizes their features and limitations, languages supported, license type, generality, whether they are specific for WSN or general, and whether they are open or closed source.

2.2.2 WSAN Pub/Sub Reference Model

Based on the insight gained from this study, we proposed a reference model for pub/sub

middleware in WSN. This model is extracted from the surveyed solution's architectures, as shown in Figure 19. A middleware layer can be added between application and operating systems layers. A complete pub/sub middleware solution should include four main components that were described earlier in middleware components section, in addition to the messaging component. Figure 19 illustrates the organization and relationship of these components. The supported services and QoS mechanisms vary from one implementation to another. For example, the routing service may be implemented within the middleware such as in Mires and PS-QUASAR or using the existing routing service such as in MQTT-S and TinyDDS. However, adding these services to the WSN platforms is very critical due to their scarce resources. As a result, it is a challenging issue to design QoS aware middleware for WSN; where it depends significantly on the application requirements. The most used platforms are TinyOS [95] and Contiki [106] operating systems over ZigBee communication protocol.

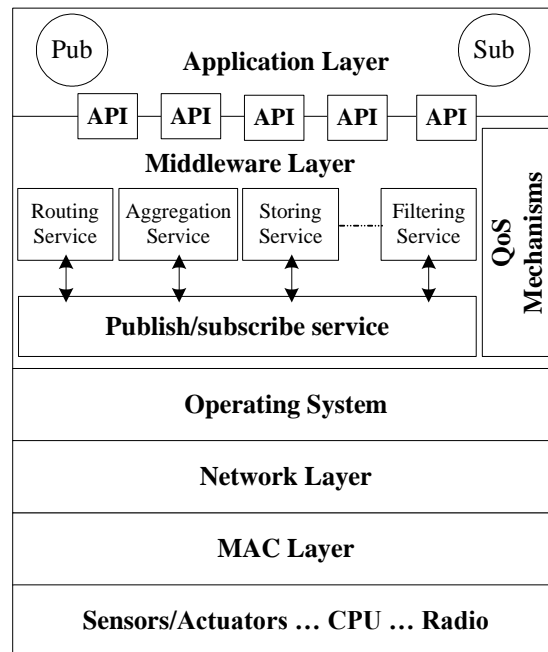


Figure 19 Pub/sub middleware reference model

Table 1 Pub/Sub WSN Solutions (D: Deadline; P: Priority; R: Reliability)

Solution	Sub Scheme	Overlay Infrastructure	Event Routing	Multiple Sinks	Actuator Support	QoS			Energy Awareness	Mobility
						R	P	D		
Directed Diffusion (2003)	Topic/content based	P2P	Sub/BCast	Y	N	N	N	N	Y	Y
Mires (2005)	Topic based	P2P	Sub/BCast	N	N	N	N	N	N	N
Quad-PubSub (2007)	Topic based	Broker	Distributed Brokers	Y	N	N	N	N	Y	N
TinyCOPS (2008)	Content based	Broker/P2P	Sub/Pub BCast	Y	N	N	N	N	Y	Y
MQTT-S (2008)	Topic based	Broker	Centralized	N	Y	Y	N	N	N	N
TinyDDS (2009)	Topic/Content based	P2P	Sub/BCast	Y	N	Y	Y	Y	N	N
PUB-2-SUB⁺ (2010)	Content based	P2P	Naming Based	Y	N	N	N	N	N	N
TinyMQ (2011)	Content based	P2P	Naming based	Y	N	N	N	N	N	N
UPSWSN-MM (2012)	Content based	Broker	Centralized	Y	N	Y	N	N	N	Y
PS-QUASAR (2013)	Topic based	P2P	Sub/BCast	Y	Y	Y	Y	Y	Y	N

Table 2 Pub/Sub WSN Solutions, Evolution and Features Summary

Solution	Test approach	Testing tools	Performance measurements	Remarks
Directed Diffusion	analytical/simulation	NS2	Avg. dissipated Energy/ Avg. delay/ distinct-event delivery ratio	Data aggregation, reverse path reinforcement, analytical analysis for data delivery cost, distributed matching process
Mires	none	None	Case Study An environment monitoring Apps / no measurements	Data aggregation, Topic advertisement, focused on facilitating WSN apps development
Quad-PubSub	simulation	JiST/SWANS	Msgs overhead/event; Hops vs subscribers;	Support for resource-awareness and shared events dissemination paths
TinyCOPS	Indoor testbed	TWIST [107]/TinyOS	Subscriptions and notifications delivery ratio / active publishers / PSLOC* / flash and RAM size	The main properties are the decoupling of communication protocols and the adaptive matching point
MQTT-S	testbed	TinyOS; Tmote and MicaZ	Just SA memory footprint (12Bytes)	Seamless integration of the WSN with traditional Networks (MQTT based)
TinyDDS	simulation / testbed	TinyOS; TOSSIM;/ SunSPOT; Solarium emulator.	PKT header overhead; Memory Footprint; Processing; and power consumption.	Standard-based solution (OMG DDS); seamless integration with access networks.
PUB-2-SUB*	simulation	Own simulator	No. of hops per even/query; No. of replicas per query; Notification delay; storage, comm., computation loads.	Content-based routing; no need for location information; less overhead than gossip routing;
TinyMQ	simulation	OPNET	Comparison with pub-2-sub in hops/query and notification delay; and repair cost (number of repaired nodes)	Adding interoperability within WSN; content-based routing without location information.
UPSWSN-MM	Outdoor testbed	HTC smart phones with Android OS; Tmote sensors with Contiki OS; Apache server (Broker)	Delay; number of delivered data; communication overhead	Supporting internet users to get sensing data anytime from anywhere; integrate WSN to internet via mobile phones.
PS-QUASAR	Simulation	Contiki (OS) TelosB motes; Cooja simulator	Energy consumption; delivery ratio of packets; delay	QoS support and high level programming; multicast support

Table 3 Simulators Used in Evaluating Pub/Sub Solutions for WSN

Simulator	Language	GUI	Generality	Open Source	License	Features	Limitations
TOSSIM [108] [109]	nesC	No	WSN	Yes	Free	*Apps ported directly to HW platform. *Bit-level simulation	*Restricted for TinyOS. *Lack decent documentations. * Add-on to support energy consumption, <i>PowerTossimz</i> [110]
COOJA [111]	Java/C	Yes	WSN	Yes	Free	*Best choice for Contiki-based WSN *Able to simulate non-Contiki nodes *easy to use and understand *Support large-scale protocols and algorithms	*Supports a limited number of Simultaneous node types. *Making extensive and time dependent simulations difficult.
OPNET [112]	C++	Yes	General	Only protocol models sources	Commercial	*Lots of protocol models including TCP/IP, ATM, Ethernet, etc. *Simple GUI to build difficult scenarios and get simulation results.	*Expensive *quite difficult to modify the protocols
NS-3 [113]	C++	No	General	Yes	Free	* Support real-time scheduling, multiple radio interfaces, and multiple channels. * Packet-level simulation.	*Lack of an application model. *Code not portable to HW. *Not scalable for WSN.
GloMoSim [114]	C/Parsec	Yes	General	Yes	Free	* supports purely for wireless networks protocols. *Using standard APIs between simulation layers. *parallel simulation support	*Less accurate in sensor networks simulations. *Code not portable to HW.
Castalia (based on OMNET++) [115] [116]	C++	Yes	General	Yes	Free	*Highly tunable MAC protocol and a flexible parametric physical process model. *Application level simulator	*Not a sensor specific platform. *Not useful for portable sensor code.
PSLOC : Physical Source Lines Of Code							

A Pub/Sub Middleware Cost in Sensor Networks

A pub/sub middleware has many benefits, as described earlier in chapter 2, such as simplifying application development and integrating sensor-based networks into access networks; also, makes the network more scalable, portable, interoperable, and flexible. However, these benefits are at the expense of sensor-based network resources. In this chapter, the cost of adding pub/sub middleware technology to WSN/WSAN is investigated. We perform an extensive simulation study to estimate the actual cost of adding pub/sub middleware to sensor nodes. Specifically, we use TinyDDS and compare it with a baseline application that is doing the same functionality without utilizing the pub/sub middleware technology.

3.1 Case Study

In this section, we describe in detail the case study that is used in the cost estimation study. Two applications are implemented, one application with middleware, and another application without middleware, called a baseline application. More specifically, TinyDDS middleware is used in middleware scenario. For the baseline application, we built a simple application that provides the same basic functionality of TinyDDS but without using pub/sub middleware technology. Both scenarios use Dynamic MANET On demand (DYMO) protocol [117] as a multi-hop underlying routing protocol.

The main function of the tested application is to collect readings from two predefined sensors, and send them to the base station. The network topology used in the evaluation is illustrated in Figure 20, where a square grid topology composed of 16 nodes is deployed in a 100x100 square meter area. The node at the upper right corner with id 15 is the base station/subscriber, and the two nodes at the bottom left corner with ids 4 and 1 are the senders/publishers, the remaining nodes are relay nodes. Thereby, the maximum number of hops nearly 3 hops, sometimes due to network congestions/failures the routing protocol selects longer paths. The network traffic load varies by changing the Inter Packet Interval (IPI) from 2 to 10 seconds. The IPI values are extracted from different simulation tests to get stable and accurate results. When we use less than 2 seconds IPI, the DYMO protocol becomes instable, where it results in very low throughput which indicates a high rate of packet loss. A simple algorithm was implemented to do the function of this tested application; where it comprises two sensors that collect the measurements of the battery voltage at sampling rate of 4Hz. The sensor then aggregates these readings locally and takes the average and send it to the base station with constant data rate based on the value of IPI, the flowchart of this algorithm is depicted in Figure 21.

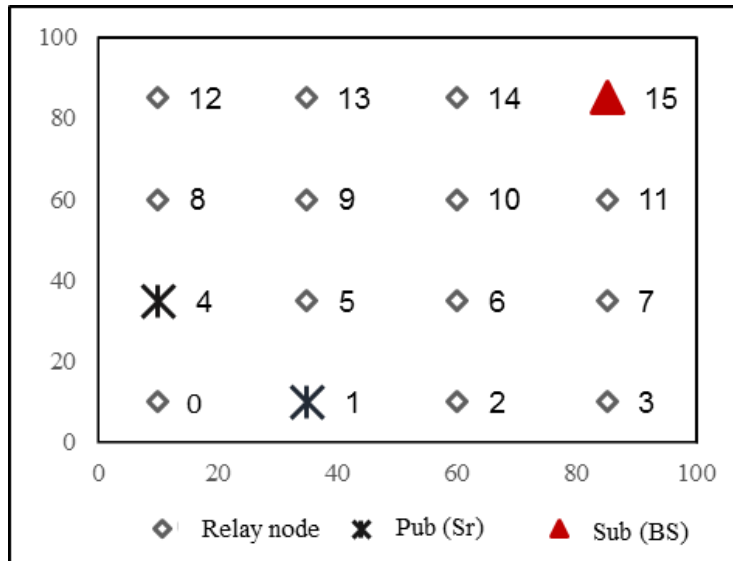


Figure 20: Case study network topology; Sub (BS): subscriber (base station), Pub (Sr): publisher (sender).

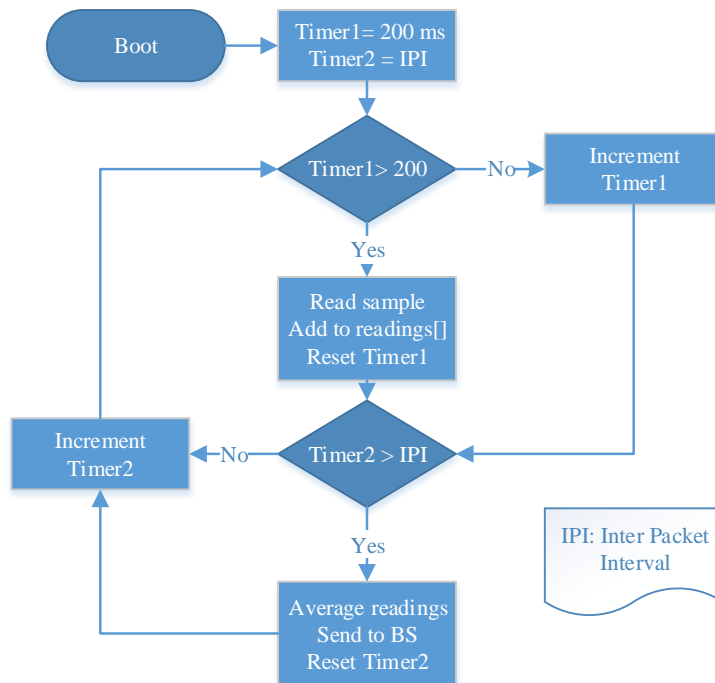


Figure 21. Basic application algorithm

3.2 Cost and Performance Evaluation

In our evaluation study to estimate the middleware cost in sensor networks, we perform several experiments using TinyOS SIMulator (TOSSIM) [118]. The main disadvantage of TOSSIM is that it does not support energy consumption measurements; therefore, we used POWERTOSSIMZ [119] to get energy consumption measurements. To perform fair comparison, the topology and simulation parameters are the same in both scenarios, i.e. baseline and middleware, as discussed previously.

Performance metrics. In our evaluation of the middleware overhead, three main performance metrics are used. 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 is 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. Although 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 total energy consumption of the whole network by taking the summation of all nodes consumption. Then, 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 footprint, which is a scarce resource in sensor devices and a critical metric in evaluating sensor applications and protocols. Both Random Access Memory (RAM) and Read Only Memory (ROM) memory footprints are measured. The number of bytes of both RAM and

ROM are measured after running the middleware and baseline scenarios. Different platforms, namely: mica2, micaz, iris, and telosb, are evaluated in terms of memory footprint.

Each data point in the results graphs is the average of ten simulation runs, and the error bars represents the standard deviation of the ten runs.

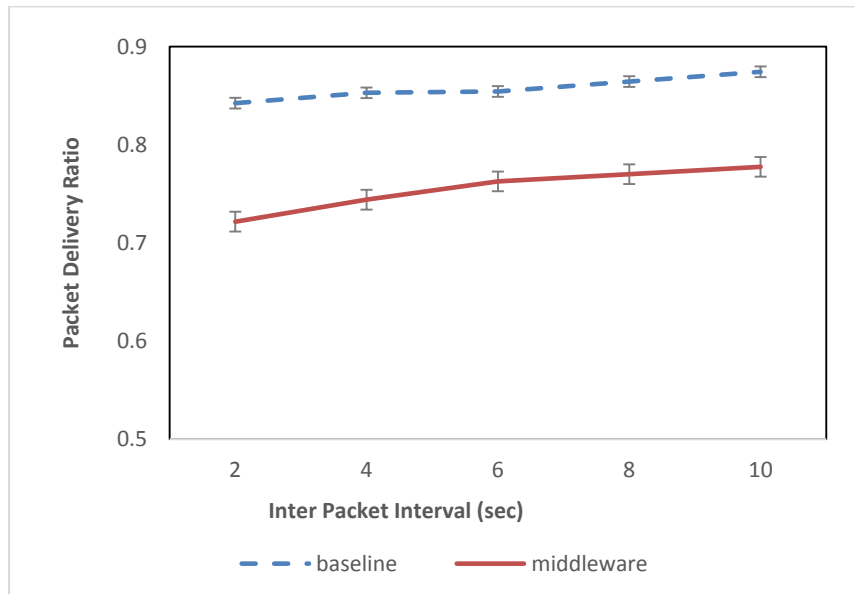


Figure 22 Packet delivery ratio comparison.

Figure 22 shows the effect of the network load on the PDR on both tested scenarios. The PDR tests the network reliability. In our test, we do not use the reliability QoS of TinyDDS for the purpose of fair comparison. It should be noted that in this study our main concern is to evaluate the middleware overhead without using its QoSs. In the baseline scenario, the PDR varies very little with the IPI, which means that the overall network load of the network is low. In contrast, the middleware scenario has larger variation 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, where it 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.

From the PDR in Figure 22, 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 23. 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 IPI = 10 sec) to 80 ms (in case of IPI = 2 sec). Intuitively, the delay decreases as network load decreases (IPI increases). However, in case of the baseline scenario the end-to-end packet delay is nearly 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 was overloaded which results in more queuing delay and might be more hops due to network congestion.

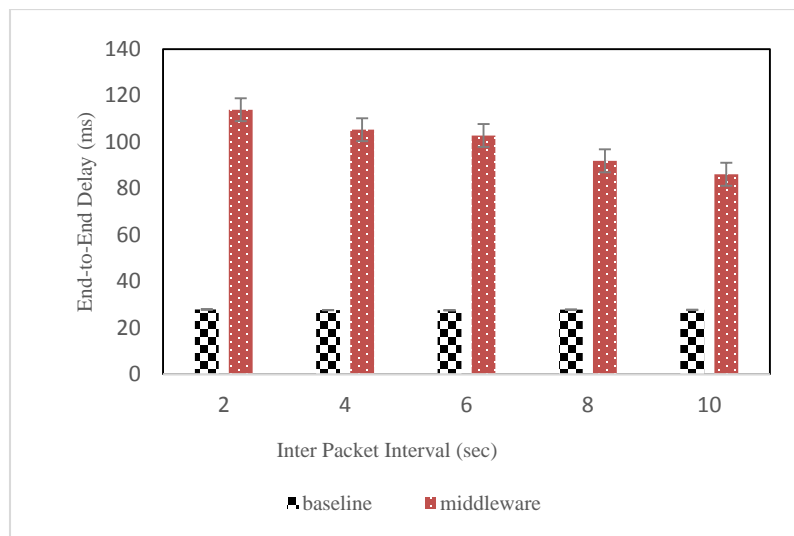


Figure 23 End-to-end delay comparison.

The memory requirements in each scenario are illustrated in Figure 24. 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 the without middleware 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 KBytes ROM, and 10 KBytes RAM, and also 1 MBytes for logs, measurements readings, etc.

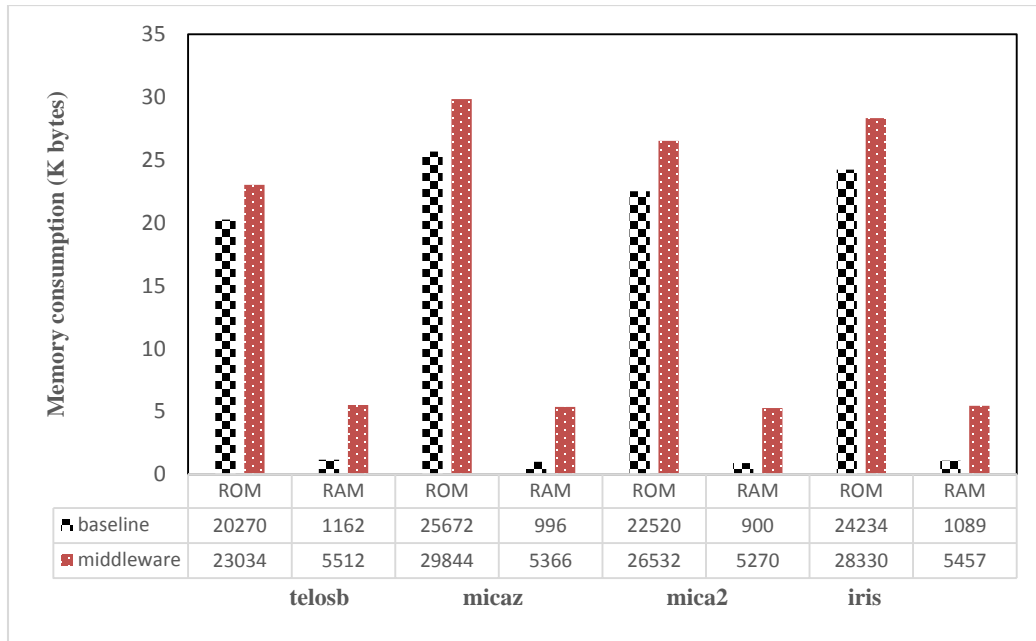


Figure 24 Memory cost comparison.

The energy consumption evaluation is conducted using the POWERTOSSIPZ tool. POWERTOSSIPZ tool assumes each node has two AA batteries with capacity of 2000 mAh. In Figure 25, the energy consumption is computed as a percentage of the average of fully charged batteries. For example, in case of IPI equals 2 seconds 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, IPI = 2 sec, the middleware consumption is higher than the baseline scenario by 37%; whereas, in case of low traffic, IPI = 10 sec, 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.

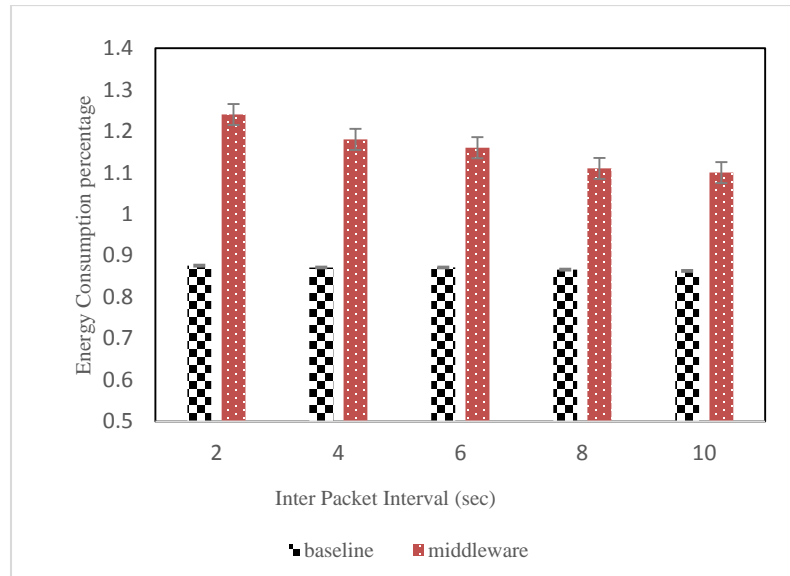


Figure 25 Energy consumption comparison

RTDDS: Reliability Protocol for WSAN

The DDS specification offers two disparate quality-of-service (QoS) levels of data reliability, namely best-effort (BE) and fully-reliable (FR). TinyDDS is a light weight and partial porting of DDS middleware to WSN platforms, specifically those with limited resources. As such, TinyDD in its current form does not support any form of reliability for data delivery. This chapter extends the DDS data reliability QoS levels by adding an intermediate level, referred to herein by the partial reliability (PR) level, and provides an implementation of the DDS reliability functions for TinyDDS. For the PR function, publisher messages are classified into either critical or not critical and then handled using the FR or BE data delivery functions, respectively. The new version of TinyDDS is called Reliable TinyDDS (RTDDS). In addition, this chapter provides a comprehensive performance evaluation of the proposed reliability functions taking into account number of hops, number of publishers, and several other network parameters.

4.1 RTDDS Implementation

In this section, a description of RTDDS protocol and its offered levels is introduced. In more details, we describe its implementation over pub/sub architecture, main components, procedure and algorithms.

RTDDS is a reliability protocol for WSN based on DDS standard. It is implemented over TinyDDS middleware, whose reliability QoS has not been implemented yet [22]. As mentioned before, DDS has two distinct reliability levels: best-effort and fully reliable. From our simulation results, we got that the cost of fully reliable QoS is very high in terms of retransmissions, e.g. around 8 retransmissions per message in case of 50% publishers and one second inter-packet interval, which consumes much energy and thus significantly reduces the network lifetime. Therefore, to suit the WSN requirements the DDS reliability QoS levels are extended by adding a new level. Intuitively, this level could be inspired from the nature of most WSN applications, where the collected data from monitoring systems is often redundant, and some of them is very important, i.e. those that exceed a certain threshold. For example, in fire or toxic gas detection systems, the sensors collect data every second or a predefined appropriate period. The data can be easily classified into reliable and best-effort based on the sensor readings, where the sensor will do in-network processing to examine the reading if it exceeds a predefined threshold, then it marks the message as a reliable message. On the other hand, if the readings are normal, i.e. do not exceed the threshold, then the sensor marks the message as a best-effort message. Consequently, RTDDS offers three reliability QoS levels that can be summarized as follows:

- **Best-Effort QoS (BEQoS):** it already exists in DDS standard, and often used for time-sensitive applications, e.g. video transmission applications. In this level, as soon as RTDDS receives a message from the application layer, it sends it only once; then the message is either successfully received or dropped. Therefore, the

reliability overhead, such as buffering, acknowledgments and retransmissions, does not exist.

- **Fully Reliable QoS (FRQoS):** it is the second level of DDS standard, and used for data-sensitive applications, e.g. file transfer applications. In this level, all messages are buffered at the publisher side until the last sent message gets an acknowledgment from the receiver. If there is a new message from the application layer while the buffer is full, the new message will be dropped accordingly. The message is persistently retransmitted until it is successfully received on the subscriber side, i.e. acknowledged.
- **Partially Reliable (PRQoS):** it is the new proposed level of DDS standard for sensor networks. In this level, the messages are classified into two types: Best-Effort and Reliable messages. The buffer at the publisher side will only be used whenever there is a Reliable message. Therefore, the Best-Effort message will only be buffered if there is a Reliable message in the buffer, otherwise it is immediately sent as soon as it is generated. In case there will be a sent Reliable message, the Best-Effort message will wait in the buffer until the acknowledgment of the sent Reliable message is received.

According to DDS standard and TinyDDS architectures, RTDDS architecture is depicted in Figure 26. As a pub/sub middleware, it includes four main entities: publisher, subscriber, pub/sub service, and the Application Programming Interfaces (APIs). DDS associates with every topic in the network two main components: Data Writer (DR), at the publisher side, and Data Reader (DR), at the subscriber side. The RTDDS basic mechanism is implemented in the DW and DR, therefore, after the modification these components are

referred to as R-DW, and R-DR, where R stands for Reliable. On R-DW, the buffering, timer, and classifier mechanisms are implemented, whereas the acknowledgment mechanism is implemented on R-DR. As shown in the architecture, the RTDDS middleware intermediates between the application and the platform details, such as TinyOS [108] protocols and Sensor/Actuator hardware. Thereby, the application can interact with the system only through the DDS API interfaces, which makes the application development easier.

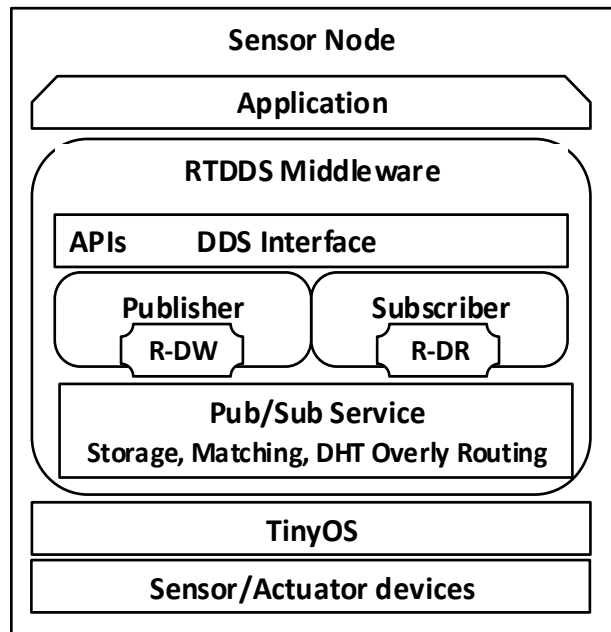


Figure 26 RTDDS Architecture

Basically, RTDDS follows the stop and wait mechanism due to its simple implementation [120] that cope with WSN requirements [121]. On the publisher side, R-DW includes three main mechanisms: buffering, timer, and classifier mechanisms. A ring buffer data structure is used to build the RTDDS buffer at the publisher side, and in our implementation the buffer size is 20 messages, each message is 20 byte. This buffer follows First In First Out (FIFO) queue discipline, where, for example, in case of FRQoS, the first message that

arrives to the R-DW component will be sent first; and if it arrives while there are some messages in the queue, it will be added to the end of the queue. If the message arrives while the queue is full, the arrived message is dropped due to buffer overflow. In the timer mechanism, the Retransmission Time-Out (RTO) is controlled. Where the timer is reset every time a message is sent, and if no an acknowledgment is received during the predefined RTO period, the timeout event occurs and a retransmission process is initiated again. In RTDDS, RTO is experimentally determined based on the available memory in TelosB platform [122] and set to 400 milliseconds. The last mechanism in the publisher side is the classifier, where the messages are classified into two types: Full Reliable (FR) and Best-Effort (BE) messages, as shown in Figure 27. Notice that this mechanism is only used in the PRQoS level. One bit is added to the TinyDDS header to be used as a message classifier, we call it reliability bit, where the application examines the readings and accordingly set this bit. If a reading exceeds the threshold, then the bit is set, which means this message is FR message. On the other hand, if the reading does not exceed the threshold, then the bit is reset, which means this message is BE. As long as there are no reliable messages in the readings the buffer is always empty. On the subscriber side, the only mechanism is the acknowledgment mechanism. In this mechanism, every arrived message is examined by checking the reliability bit. If this bit is set (FR message) then a corresponding acknowledgment message is generated and sent back to the message origin. Otherwise, if the bit is reset, the BE message is silently received without any response to the message origin.

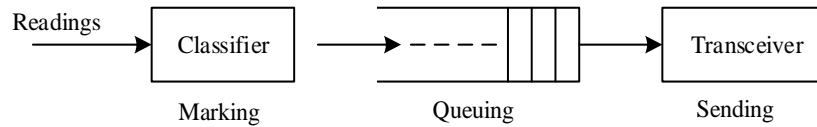


Figure 27 RTDDS Classifier on the publisher side

Figure 28 illustrates the flow chart of the main RTDDS algorithm with FRQoS level. Since we are using ring buffer and FIFO queue discipline, the “buffer in” means adding the new message to the end of the buffer, and the “buffer out” means sending the first message from the front of the buffer; and the “buffer remove” means removing the first message at the front of the buffer, since it has already successfully received on the subscription side. Therefore, in case of retransmitting a message we just recall “buffer out” command. The buffer out command either resends an old message when the timeout event is fired, or sends a new message when the corresponding acknowledgment of the sent message is received. In the case of PRQoS, the subscriber sends its interest to the middleware service with a certain threshold, hereby the classifier at the publisher side classifies the messages into BE and FR messages based on the required threshold. For instance, in case of one sample exceeds a certain specified threshold, the publisher will mark this message as a FR message; otherwise it will mark it as a BE message, Figure 29 shows a simple pseudo code for the algorithm of PR QoS of RTDDS, which is implemented in both R-DW, and R-DR components. This algorithm is integrated into the main algorithm in Figure 28. In PRQoS algorithm, we use a *wait* variable to wait for the acknowledgments of the FR messages. Consequently, any message arriving at the R-DW, whether it is a BE or FR message will be buffered until the acknowledgment of the sent FR message is received. Thereby, we ensure in-sequence data delivery service, since the BE messages cannot be sent until the all front FR messages are sent. In DDS each data writer and reader is associated with a

particular topic. Therefore, each topic in RTDDS can be associated with different QoS level, and each subscriber can also request a different QoS level. For example, in one WSN scenario there might be several subscribers and each of which requested a distinct QoS level, best-effort, fully reliable, or partially reliable QoS.

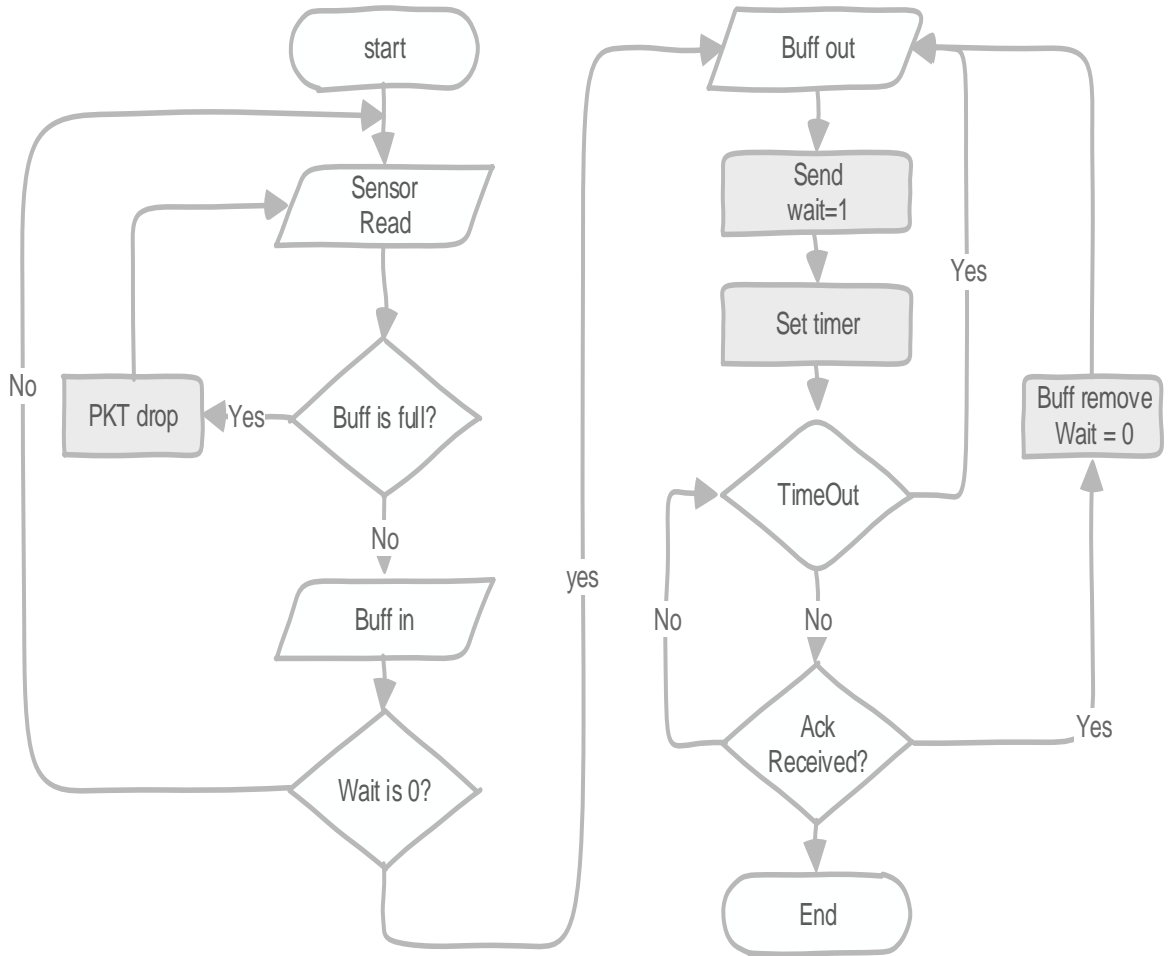


Figure 28 The reliability algorithm of RTDDS with FRQoS level

Algorithm1: Data-Writer Partial Reliability QoS

```
1:  If msg is BE and wait is False
      then msgSend;
2:  Elseif msg is BE and wait is True
      then BufferInsert;
3:  Elseif msg is FR and wait is False
      then msgSend;
      wait=True;
4:  Elseif msg is FR and wait is True
      then BufferInsert;
```

Figure 29 PRQoS level algorithm

4.2 Performance Evaluation

In this section we extensively evaluate the proposed reliability protocol RTDDS. The evaluated factors that have impact on RTDDS performance are: Retransmission Time Out (RTO), Number of hops, data rate or Inter-Packet-Interval (IPI), number of publishers, and the percentage of reliable messages in case of PRQoS. The impact of these factors is evaluated using the following performance metrics:

- ***Packet Delivery Ratio (PDR)***: the main performance measurement of a reliability protocol. It is calculated by dividing the total number of successfully received messages at the subscriber side by the total sent messages from the publisher side. For an ideal reliability protocol this metric should be equal to one for all scenarios.

- ***End-to-End Delay (EED)***: it is measured from the moment of sending or publishing data on a publisher side until it is successfully received on a subscriber side. Therefore, it includes the buffering time at the publisher side, which is the major effect on the delay, and also the transmission, propagation, and receiving time at the subscriber side; the retransmission trails also included.
- ***Dropped Message Ratio (DMR)***: this metric is related to the buffering at the publisher side, where it calculates the dropped messages due to the buffer overflow. This happens when a message arrives to the data writer while it is still waiting for an acknowledgment and the buffer is full. Thus, the DMR is calculated by dividing the total number of dropped messages due to buffer overflow by the total number of sent messages by the application layer.
- ***Retransmissions per Message (ReTx/Msg)***: this metric represents the cost of successfully received messages in terms of number of retransmissions. Each sent message might be successfully received from the first sending time or it may need to be retransmitted several times until it is successfully received. This metric is calculated by dividing the total number of retransmissions by the total number of successfully received messages.
- ***Redundant per Message (Rd/Msg)***: this metric indicates the efficiency of the protocol in terms of redundant messages received at the subscriber side for the same sent message. It is calculated by dividing the total number of redundant messages, excluding the first received message, by the total number of successfully received messages.

- **Energy consumption:** this metric is very important in WSN since energy is scarce and it determines the network lifetime. It is measured as the voltage drain by the network nodes from the moment the network is initiated until the last message received from the last alive node in the network.
- **Memory footprint:** this metric is measured as the number of bytes consumed by the RTDDS code, when it is uploaded to TelosB platform. Both RAM and ROM memories are considered in evaluating this metric.

4.2.1 Experiments setup

Several experiments were conducted to evaluate the RTDDS performance. These experiments are divided into simulation and empirical experiments. In both types, RTDDS is compared against TinyDDS, where TinyDDS is represented by the BEQoS. The empirical experiments are conducted using TelosB motes. While Table 4 includes all the common simulation parameters, Table 5 specifies the variable network parameters and their values in each used scenario. Three main scenarios are used in simulation experiments: two for FRQoS, where RTO, IPI, number of publishers, and number of hops are examined, and one for PRQoS, where reliability percentage factor is examined. RTDDS is tested over two platforms, one by TOSSIM [109] simulator, a micaZ mote platform, and the other by a prototype that is downloaded over TelosB motes. We use static routing for multi-hop scenarios, and the radio model is based on Chipcon CC2420 model [123]. For more details on the experiments' simulation setup, refer to Table 4. Each data point in the results represents the average of ten times of simulation runs. In addition, the standard deviation of the ten runs is represented by the error bars in the results' charts.

Table 4 Simulation setup

Parameter	Value
Topology	Squared grid
Area	100 X 100 Meter ²
Number of Nodes	50
Simulation time	500 seconds
Radio model	Chipcon CC2420 [123]
Mote platform	micaZ
Data rates	60, 30, 20, 15, 12 Msg\Minute
Number of publishers	1, 5, 10, 15, 20, 25
Message size	20 bytes
Maximum hops	10
RTO	400 milliseconds
Percentages of Reliable messages	0, 20, 40, 60, 80, 100 %
Runs per results' data point	10

Table 5 The three main scenarios in the simulation study

Scenario	Examined factors	Performance metrics	No. of publishers	No. of hops	IPI (sec)	RTO (ms)	Reliability level
FRQoS-RTO	RTO, No. of hops	PDR, ReTx/Msg, EED, DMR	2	1, 2, 3, 4, 5	1	200, 400, 600, 800, 1000	FRQoS
FRQoS-IPI	IPI, No. of publishers	PDR, Rd/Msg, EED, DMR	1, 5, 10, 15, 20, 25	Max 3	1, 2, 3, 4, 5	400	FRQoS
PRQoS	Reliability percentage	PDR, No. of ReTx, EED, DMR	5	Max 3	1, 5	400	FRQoS, PRQoS, BEQoS

4.2.2 Full Reliability QoS Results

Since FRQoS is the level causes the largest protocol communication overhead, almost all the factors that affect RTDDS performance are evaluated in this level. Moreover, we experimentally adjust the RTO according to this level. Consequently, the performance of the other two levels, i.e. PRQoS, and BEQoS, would be less effected by the different network parameters. That means, under the same examined conditions here, the other levels perform better than FRQoS level in terms of EED, ReTx/Msg, Rd/Msg, and DMR; and, in return, FRQoS is the best in terms of PDR.

Before starting the simulations, an improvement is added to RTDDS to minimize the significant effect of the Co-Channel Interference (CCI) on RTDDS protocol. Figure 30 shows the significant effect of the CCI on the PDR of the BEQoS level. To reduce this effect, we use a simple algorithm for Interference-Free Scheduling (IFS) at the middleware layer, in which each set of adjacent nodes are sending at different times, i.e. Time Division Multiple Access (TDMA). Using IFS algorithm improves the PDR of the BEQoS level nearly 3.5 times. In contrast, the FR level shows more robustness against CCI, where the PDR almost the same in both cases, with and without IFS. However, in the cost of retransmissions and delay, in case of five IPI it incurs about 1.9 times the number of retransmissions of with IFS, and the delay of with IFS is nearly 2.4 times less than in case of without IFS (NIFS). Also, notice that the PDR of the BEQoS level almost the same in both cases of one and five IPI, which is because the nodes in case of five IPI stay not active for almost four seconds and then send at the 5th second, consequently, the channel contention would be almost the same in both cases of one and five IPI. Finally, in terms of PDR the FRQoS is more robust than BEQoS, which is because FRQoS level persistently

deliver the data to the receiver side. That can be deduced from the error bars showed in the figure, where they are much higher in case of BEQoS level.

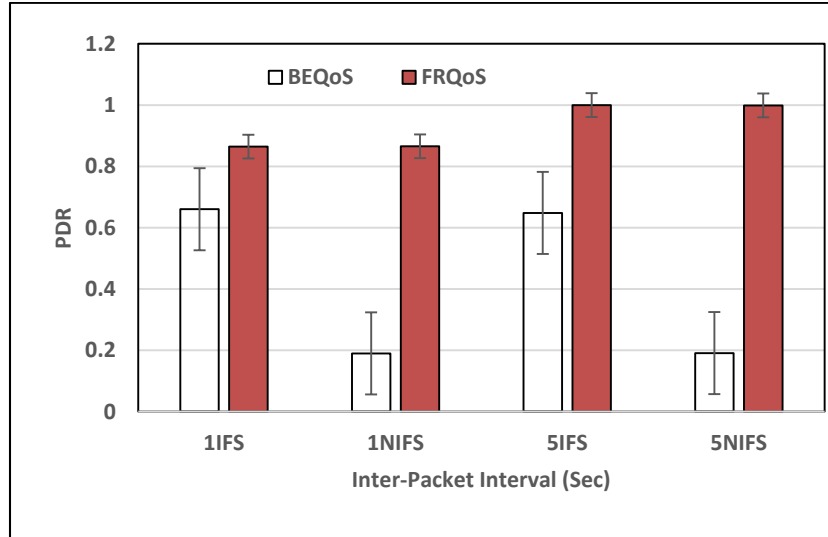


Figure 30 Interference effect on the performance of RTDDS

In FRQoS-RTO scenario, we use two publishers and one subscriber (BS) with different number of hops and different values for RTO, as described in Table 5. Figure 31 depicts the RTDDS performance versus number of hops and also versus different RTO values. The hops are started from one to ten hops, however, it is worth mentioning here that the smallest number of hops is two hops, i.e. one to the rendezvous node where the matching process and publication routing are conducted, as mentioned earlier in Chapter 2, and one to the interested subscriber. For instance, if one publisher is away from the interested subscriber by six hops, that means three to the rendezvous node and three to the interested subscriber. In addition, the RTO values range from 200 to 1000 milliseconds (ms), where 200 ms is the minimum Round Trip Time (RTT) of five hops distance in our testing environment. That means, the minimum time required from the publisher to wait for the acknowledgment is equal 200 ms, in case of five hops distance between the publisher and BS. And the upper bound is 1000 ms because the data rate in this test is one message per second, thus if the

timeout is higher than the data rate it causes buffer overflow, which leads to system instability.

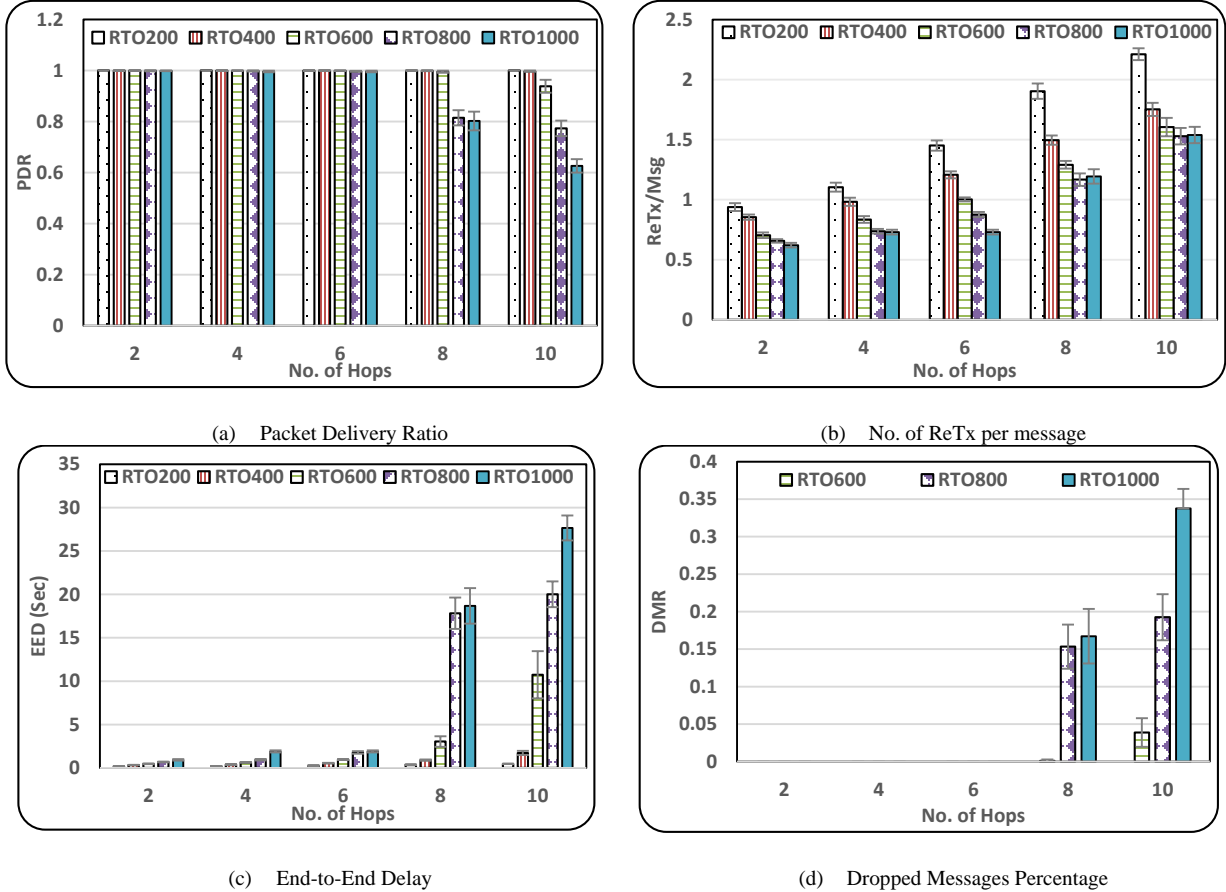


Figure 31 The impact of RTO and No. of hops on RTDDS performance

Since the packet loss is the most important performance metric in reliability protocols, we give it the highest priority in our selection of RTO. In Figure 31, part (d), we can see that there are no dropped messages in case of RTO 200 ms and RTO 400 ms, even in the worst case, i.e. ten hops. From the same part, it shows that RTDDS with FRQoS level is robust until six hops whatever the timeout is. That is because in case of six hops the PDR is 100% and the DMR is zero for all the cases of RTO. In addition, in part (d) it also shows that until the six hops case the delay cost is in the range of milliseconds. In general, the four parts of the figure show that the RTDDS performance degrades as the timeout increases,

since it goes to the system instability state, as discussed. Herein, we have two choices, 200, and 400 ms to be used as our RTO of RTDDS. From part (b), the ReTx/Msg cost is lower in case of 400 ms than 200 ms, and at the same time both of them almost have the same PDR in part (a). As a result, we selected 400 ms as RTDDS Retransmission timeout for the rest of our tests.

In FRQoS-IPI scenario, RTDDS with FRQoS level is evaluated under heavy network conditions such as increasing number of publishers and data rate, the results are shown in Figure 32. The number of publishers is increased until it reaches 50% of the network nodes, and network data rate is increased from one message per five seconds to one message per one second. The reason why we chose this range is that we experimentally decreased the data rate until we got zero messages dropping in case of 50% of the number of publishers. Wherein, the minimum data rate to get this result was one message per five seconds, as shown in part (d). In this part, the worst case is with one sec IPI and 50% number of publishers, where the DMR is around 60% and that is due to the high collision rate because of the heavy network load. From the same part, we can see that the only guaranteed scenario to ensure reliable data delivery service is the five sec scenario, where there is zero dropped messages in all cases of the number of publishers. Therefore, we can deduce that RTDDS with FRQoS is more suitable for applications that have a time-sensitivity response time not less than five seconds. Thus, the maximum delay of the five seconds scenario would be, as shown in part (c), around five seconds with 100% PDR, as shown in part (a). As an alternative measure for cost, we use redundant per message instead of retransmissions per message, as shown in part (b), which is around 60% redundant messages in case of five sec scenario. In general, from the four parts, the results are intuitive where the cost in terms of

EED, DMR, and Rd/Msg increases as the network load increases in terms of data rate and number of publishers, whereas the performance in terms of PDR decreases.

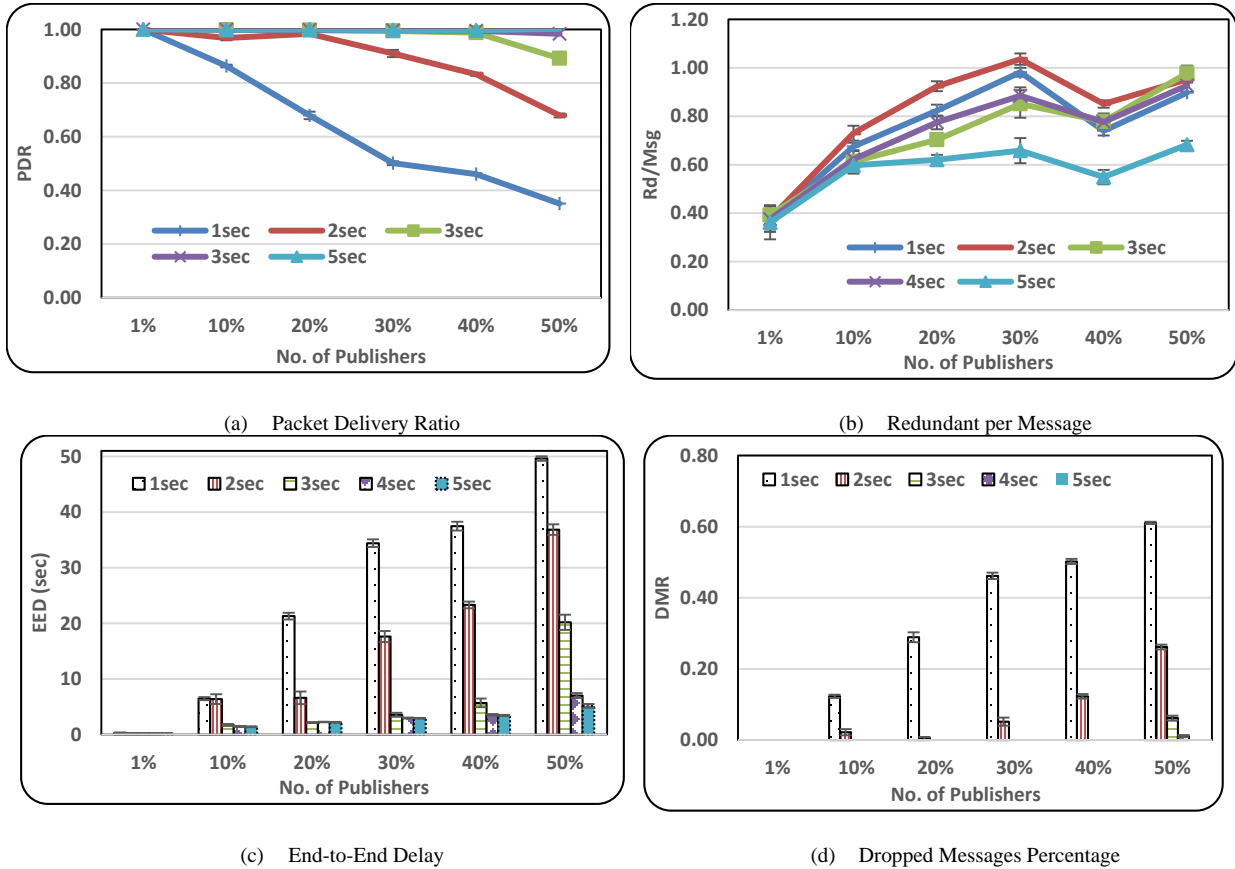


Figure 32 The impact of number of publishers and data rate on RTDDS performance

In the previous analysis we targeted the time-sensitive applications or soft real-time applications, since the sensor updates are in the order of few seconds. In Table 6, RTDDS is validated by simulation that it works perfectly in the non-time sensitive applications, e.g. with data rate of one packet per minute or slower. In this test, 50% of the sensors send one packet every one minute to the base station. As shown in the table, the RTDDS cost in terms of delay, retransmissions, and redundant messages is significantly reduced. Where the delay is minimized from around five seconds to 243 milliseconds; also, the ReTx/Msg

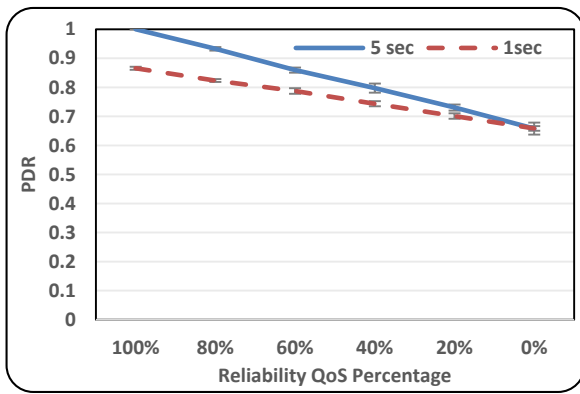
are reduced from nine messages to one message. Moreover, Rd/Msg minimized from 68% to 42%, which means that less than the half of the transmitted messages would get redundant in case of data rate of one Msg/Minute.

Table 6 RTDDS performance with high and low data rates

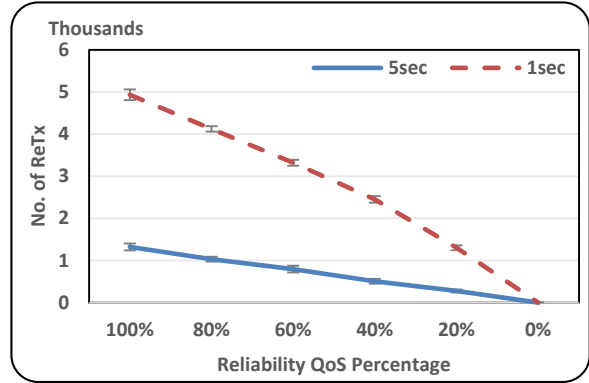
IPI	SimTime	# MSGs	PDR	Delay	ReTx/PKT	Dropd	Rd/PKT
5	500	2500	1	5172	9.22	0	0.68
60	7200	3000	1	243	1.18	0	0.42

4.2.3 Partial Reliability QoS Results

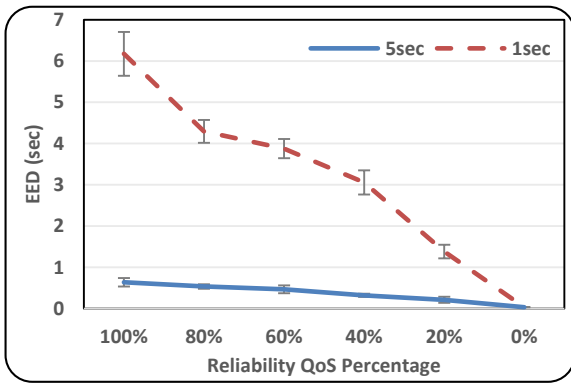
In PRQoS scenario test, we use the scenario of five publishers are sending to the base station (subscriber) with two different data rate, and the network maximum hops are six from the publisher to the base station. The worst case represented by one second IPI, where there would be message dropping, due to buffer overflow. The second scenario is the zero message dropping scenario, which is represented by five seconds IPI, since it has been tested experimentally and there was no message dropping until 50% of the network are publishers. As discussed previously, in PRQoS the first two levels are employed to work together, namely BEQoS and FRQoS levels. To observe the effect of different levels of PRQoS on the protocol performance, we control the published messages in which the percentage of FR messages is 0%, 20%, 40%, 60%, 80%, and 100% from the total sent messages. Notice that 0% represents BEQoS level and 100% represents FRQoS level. Thus, this test can be considered as a comparison between the three QoS levels offered by RTDDS.



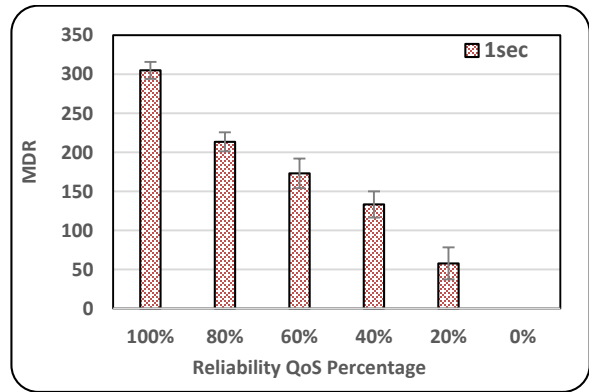
(a) PDR with five and one second IPI



(b) No. of RxTx per message for 2495 messages



(c) End-to-End delay (second)



(d) No. of dropped messages

Figure 33 Partial Reliability QoS results with five and one seconds IPI and five publishers

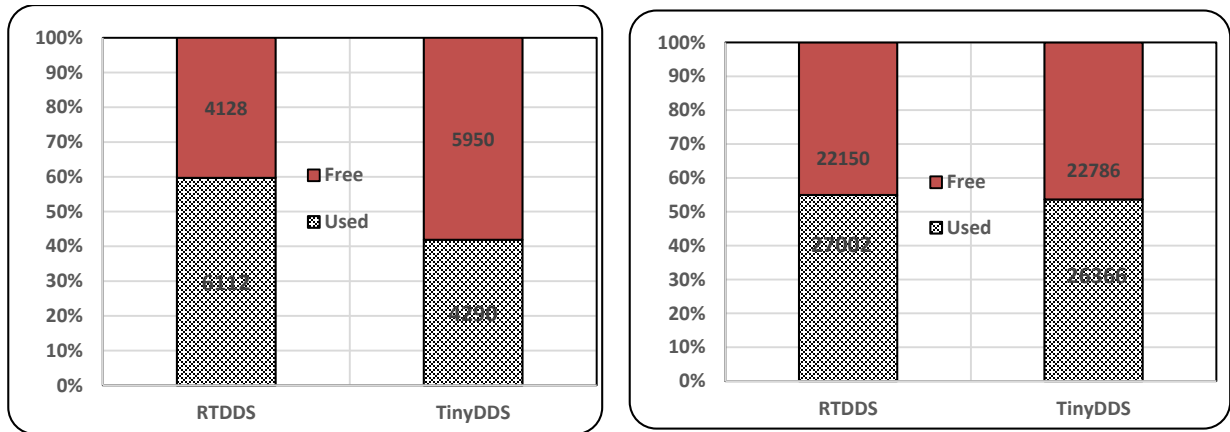
As the previous results, Figure 33 depicts the comprehensive results of this test, including PDR, No. ReTx, EED, and the DMR. In part (a), intuitively, as the number of FR messages decreases the PDR decreases. That is expected because the number of unguaranteed messages increase as we increase the BE messages, also the effect of the packet dropping due to network conditions become more observable. At 0% PRQoS, which exactly means BEQoS, we can observe that there is no difference between one and five seconds IPI. That is because the packet dropping due to the channel contention is the same in one and five seconds' scenarios. That means, in both scenarios the sending period is almost one second for all the publishers, and this one second could be the first or the fifth, where the remaining

four seconds the sensors stay in the sleep or inactive mode. This might be improved by extending the IFS (Interference-Free Scheduling) based on the IPI of the application, i.e. in case of five seconds, the transmissions would be distributed over the whole 5 seconds. In part (b), due to the huge traffic of applying reliability QoS and the fast publication rate in case of one second IPI, massive packet dropping occurs and thus it is reflected by the number of retransmissions. However, it is very important to notice that both the 5000 ReTxs and 1000 ReTxs in case of FRQoS are corresponding to two ReTxs/Msg, that is because the total number of sent messages is 2500 and 500 respectively. For real-time systems that have response time sensitivity less than one second, applying full reliability required huge amount of resources in terms of processing, memory, bandwidth, and power. However, since the number of retransmissions decreases from around 5000 ReTxs to 1000 ReTxs, this emphasizes that RTDDS is very suitable for the applications that have response time sensitivity around one minute or more. This observation can also be supported by the result of part (c), where in the case of five seconds the end-to-end delay is in the order of milliseconds, even when we use 100% PRQoS or FRQoS. Finally, part (d) shows the dropped messages at the publisher side because of buffer overflow. As shown in the figure, only the scenario of one second IPI is illustrated because there is no message dropping in five seconds scenario. The number of dropped messages almost linearly decreases with the PRQoS percentage decreasing. In FRQoS level, nearly 12% of the messages are dropped, whereas in 20% PRQoS, nearly 2% of the messages are dropped. Therefore, it is obvious that in case of partial reliability QoS the reliability protocol is significantly improved. Besides, PRQoS level is often used in sensor applications, where most of the data is redundant unless a few readings that exceeds the predefined threshold.

4.2.4 Memory and Energy Consumption Results

For more accurate and realistic measurements, we develop an RTDDS prototype version and install it on TelosB mote platform. In this test, RTDDS with FRQoS and normal TinyDDS are compared regarding memory and energy consumption. The test scenario includes four motes: Base station, Rendezvous, RTDDS, and TinyDDS nodes. These motes are deployed in indoor environment. Notice that, in terms of energy consumption, we can consider this test as a comparison between FRQoS and BEQoS, since TinyDDS default QoS is a BEQoS level. The publishers of RTDDS and TinyDDS nodes send data with one message per second rate to the base station through the rendezvous node. The message size is 20 bytes, and it is acknowledged by the base station in the RTDDS case. The RTDDS and TinyDDS nodes are supplied by AA Energizer batteries, which means that each one has an initial voltage of around 3V.

In the memory test, the RAM and ROM occupied space is computed as a percentage of the free and used memory where TelosB RAM is 10 Kbytes and ROM is 48 Kbytes. Figure 34 shows the results of the memory space occupied by RTDDS and TinyDDS. Part (a) represents the RAM usage, wherein RTDDS and TinyDDS occupy around 60% and 40% respectively. Thus, the difference is 20% more by RTDDS, which is because of the buffer at the publisher side and the control variables in both sides such as wait and timer variables. In conclusion, the RAM still has 40% after adding reliability protocol to TinyDDS, which makes it extremely efficient and applicable. Furthermore, in part b, the ROM test supports this conclusion, where the difference even much less than in the RAM in which RTDDS implementation increases the ROM by only around 5% compared to TinyDDS.



(a) RAM usage by RTDDS and TinyDDS

(b) ROM usage by RTDDS and TinyDDS

Figure 34 RTDDS and TinyDDS Memory consumption based on TelosB platform

To measure the Network Life Time (NLT) the network continuously works until the receiving of last message just before batteries death. Figure 35 shows the results of RTDDS and TinyDDS NLT when data acquisition is continuously being performed with one message per second data rate. According to TelosB reference [124], the minimum voltage for the mote to work properly is 1.8 V as illustrated in the figure. However, both RTDDS and TinyDDS motes work until it reaches 1.53 V. The result shows that the NLT of RTDDS and TinyDDS are 5.5 and 6.25 days respectively. Of course this is too short because our test is conducted under intensive data rate, whereas in real world applications the duty cycles are much less than that, and energy saving modes are also used, i.e. sleep, and deep sleep modes. As a result, the real world NLT would be extended to months or even years. Moreover, we can observe that RTDDS, which is working in FRQoS level, energy consumption (represented by voltage drain) is more than TinyDDS due to the extra traffic used as acknowledgments, and more processing for reliability mechanisms. Further, the difference between TinyDDS and RTDDS increases almost linearly with time. However, it is important to consider that our test is nearly a perfect environment since there were

almost zero retransmissions in case of RTDDS. It is also worth noting that the total data received by RTDDS, and TinyDDS are 471561, and 517322 bytes respectively. Accordingly, the volts per bit for both scenarios are $1.94832E-08$, and $1.77597E-08$ respectively. Given this information, this result is important, since it would be used for energy consumption or NLT estimation for RTDDS or TinyDDS middlewares. Finally, the result shows that RTDDS is applicable and efficient in terms of energy/memory consumption.

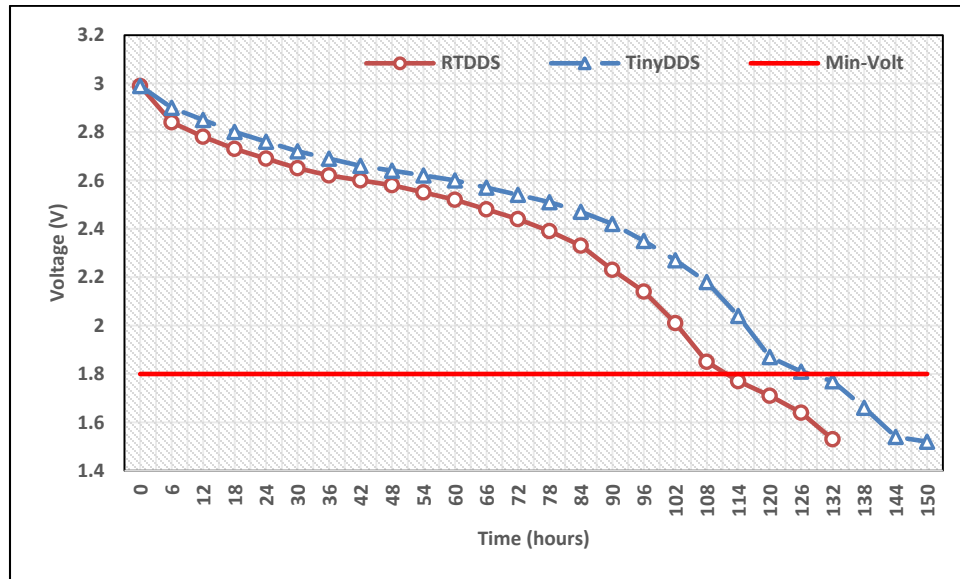


Figure 35 Energy consumption of RTDDS (FRQoS level) and TinyDDS

CHAPTER 5

ONLINE ENERGY MODEL

TinyDDS is implemented over TinyOS code, therefore, the main challenging issue in this research was how to use *TinyOS SIMulator* (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.

One of the most well-known and accurate simulators for wireless sensor networks is TOSSIM [109] [108], 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 have been developed to tackle this problem by integrating energy measurements tools into TOSSIM. These extensions are: POWERTOSSIM [125] and POWERTOSSIMZ [110], where the difference between the two is that 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 depends on the energy measurements from the micaZ datasheet [126].

The main limitations of the POWERTOSSIP and POWERTOSSIPZ are: (1) they support only the mica series platforms, while they do not support telos platforms; (2) they do not support online energy measurements since they compute the final energy measurements after the simulation run and based on post processing process that uses the energy model of the used platform. The second limitation is very important for any energy aware simulation study, because in such protocols the energy level of the network nodes should be known during the simulation to take the proper action according to the energy readings. Therefore, one of the challenging issues in this work was 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 TOSSIP components. Furthermore, we validate our model by comparing our results with the previous work PWOERTOSSIPZ.

5.1 Online Energy Model Description

Unlike POWERTOSSIPZ, in Online Energy Model (OEM) we only focus on the Radio and MCU components, 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 TOSSIP 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 our online energy model implementation are the *TossimPacketModel* component which is corresponding to the Radio unit, and *SimSchedulerBasic* which corresponds to the MCU

unit. Figure 36 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.

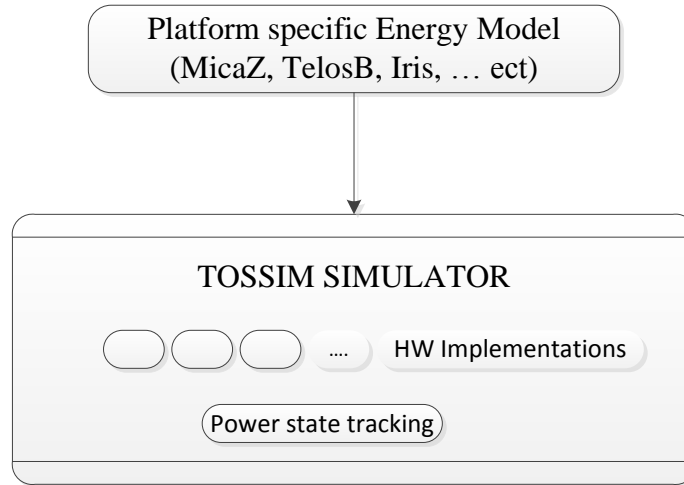


Figure 36 Online Energy Model Architecture

5.1.1 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 *Splitcontrol*. 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 5.1, where the Δt represents the state duration (receiving, sending or sleeping), and V represents the used voltage, which is approximately 3 V, and I_{state} 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 7. The OEM Radio algorithm is shown in algorithm 5.1.

$$EC_{Radio} = \Delta t * V * I_{Pstate} \quad (5.1)$$

Algorithm 5.1: Online Energy Consumption of the Radio

```

while sim ranning do {
    while RoundTimre not fired do {
        if (send.start || receive.start)
            eventstamp = simtime;
        if (send.done || receive.done) {
            duration = simtime-eventstamp;
            ActiveTime +=duraiton;
            if(send.done) ECs += duration * V * Itx;
            else ECr += duration * V * Irx;
        }
    }
    IdleTime= SimTime - ActiveTime;
    ECi += IdleTime * V * Iidle;
    send (ECs,ECr,ECi);
    reset RoundTimer;
}

```

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.

Note: V, Itx, Irx, and Iidle are fed into the simulator before running based on the tested platform.

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 [127]. Hereafter, these variables can be easily accessed by any component in the simulator during the simulation run.

Table 7 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

5.1.2 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 5.2 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 Table 8. The main states that we use in our tests for MCU were Active and Idle states as illustrated in algorithm 5.2. As described earlier in this chapter, the MCU power state tracking code is integrated with *SimSchedulerBasic.nc* component, specifically in the scheduler.runNextTask event. The main condition used in the MCU algorithm 5.2, if the

scheduler has no tasks, then the MCU in the idle state; otherwise it is in the Active state.

The OEM of MCU algorithm is shown in algorithm 5.2.

$$EC_{MCU} = \Delta t * V * I_{Pstate} \quad (5.2)$$

Algorithm 5.2: Online Energy Consumption of the Microcontroller

```
while sim running do {  
    while RoudTime not fired do {  
        duration = SimTime - PrevStateTime;  
        if (PrevState == Active)  
            ECactiveMCU= duration * V * IactiveMCU;  
        if(PrevStateMCU == Idle)  
            ECidle = duration * V * IidleMCU;  
        if (nextTask == noTask) {  
            PrevState = Idle;  
            PrevStateTime = SimTime;  
        } else {  
            PrevState = Active;  
            PrevStateTime = SimTime;  
        }  
    }  
    Send(ECactiveMCU, ECidleMCU);  
}
```

Table 8 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

5.2 Simulation and Validation

Since the last extension for TOSSIM that enable energy measurements was the POWERTOSSIMz (PTZ), our validation will use PTZ results to be compared with our OEM results. 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 with our 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 37 and Figure 38 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 9 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.

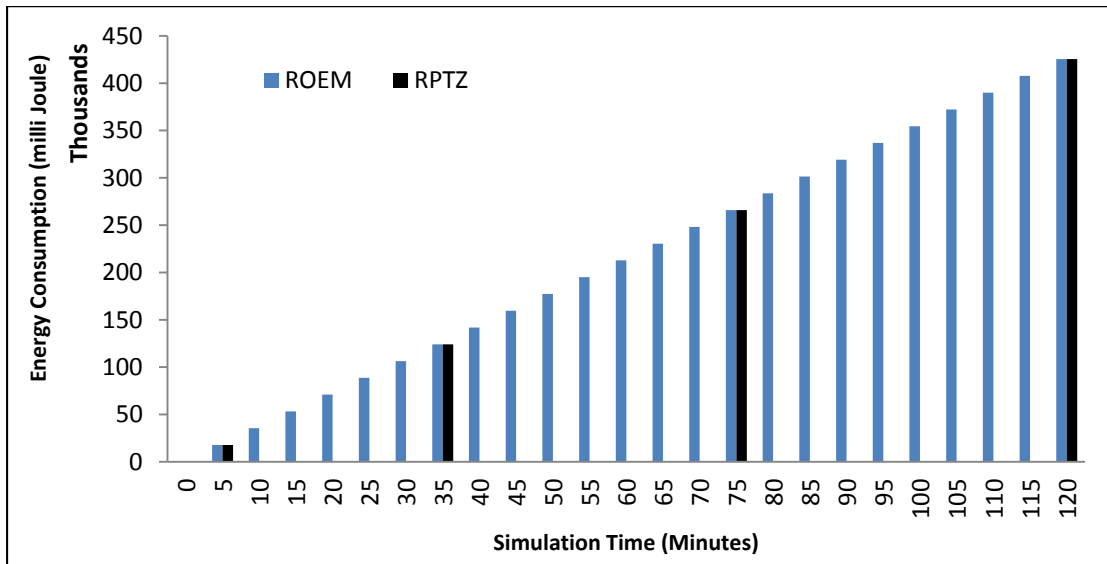


Figure 37 Energy consumption of the Radio Component

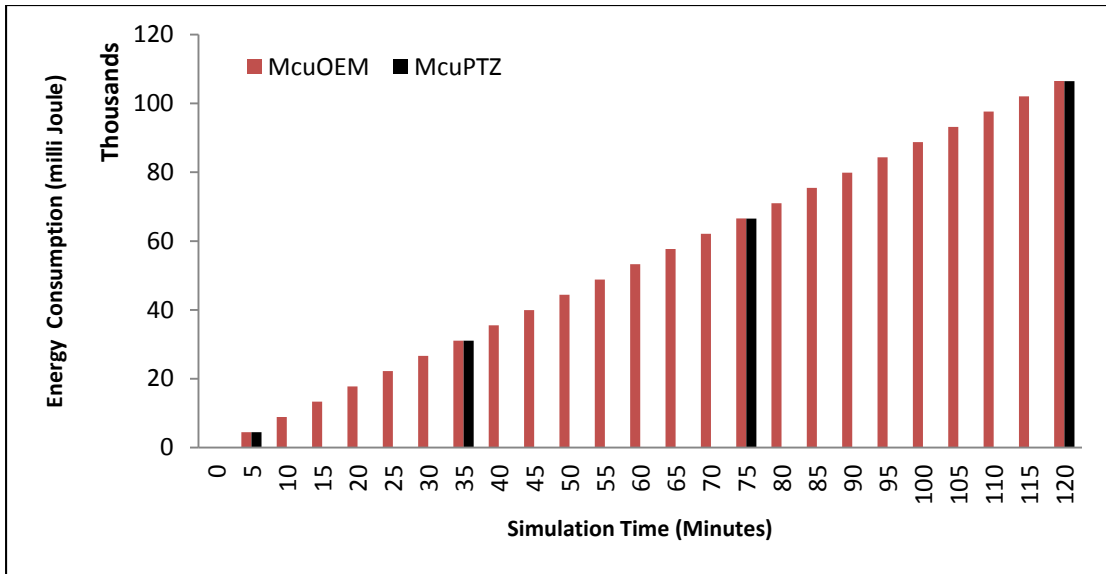


Figure 38 Energy Consumption of the MCU component

Table 9 The OEM and PTZ validation comparison

Simulation Time (sec)	Radio (mJ)			MCU(mJ)		
	PTZ	OEM	Error %	PTZ	OEM	Error %
5	17776.1	17727.92	-0.00271	4449	4454.685	0.00128
35	124084.6	124095.4	8.71E-05	31056.3	31076.92	0.00066
75	265907.2	265918.7	4.33E-05	66552.1	66574.06	0.00033
120	425459.3	425470	2.51E-05	106485.3	106508.5	0.00022

CHAPTER 6

BROCKER-LESS TinyDDS

In this chapter we describe two alternative solutions that can be used to eliminate the Rendezvous/Broker node in TinyDDS. These methods are called Broker-Less TinyDDS (BLTDDS) and Hybrid TinyDDS (HyTDDS). The two methods are compared with the original method of TinyDDS, which we call here Default TinyDDS (DefTDDS). Throughout this chapter we will call the Rendezvous/Broker nodes as Rendezvous Node, abbreviated as (RN).

6.1 Proposed Solutions

In any pub/sub system, the participant has two main phases: (1) Discovery phase and (2) Data Dissemination phase. As soon as the node joins the network it starts the discovery phase by sending subscription messages to the middleware until it is recognized and then switch to the data dissemination phase, where the middleware starts sending the data of interest to the joining node/subscriber. From the survey study we can see that the main routing methods of the subscription messages or data (publications) are either broker-based or broker-less (P2P). TinyDDS uses the broker-based methods in routing its subscription and publication messages. We argue that this centralized method is not suitable for the function of WSN platforms, because it will form a bottleneck that will rapidly exhaust the node energy, and eventually ends the network life time while the network still has adequate residual energy. Therefore, in this chapter we study two alternative solutions: (1)

Broker-Less TinyDDS (BLTDDS); and (2) Hybrid TinyDDS (HyTDDS). The BLTDDS has been used in several solutions [8] [19]. In this method, the subscriber broadcasts subscription messages to all nodes in the network, where the matching process is conducted at the publisher side. We propose a new hybrid method that can avoid the flooding overhead of BLTDDS method and mitigate the bottleneck problem of Default TinyDDS (DefTDDS). In this chapter the main assumption is that the middleware knows all the publishers of the networks at the deployment time. In the following sections we elaborate more on the pub/sub routing process for the three methods.

6.1.1 Default TinyDDS

In DefTDDS, one Rendezvous Node (RN) for each topic is assigned; where the publishers and subscribers of that topic will meet. In Discovery phase, this node receives the related publications and subscriptions. The RN address is obtained at the end nodes, i.e. publishers and subscribers, using the hashing function in equation 6.1; where the *Topic identification* and *max Topic* numbers are known before the network deployment, as they are part of the middleware core's parameters.

$$RN_{address} = Topic_{id} \text{ Modulus } MaxTopics \quad (6.1)$$

In Data Dissemination phase, due to the memory limitation of the sensor/actuator devices, the RN node uses volatile QoS, i.e. new publications are not saved in the RN memory, and as soon as the new publication is received, it is forwarded to the all interested subscribers that are registered in the RN database, and then it is deleted immediately, and so on. If there is more than one subscriber for a single topic, the RN receives single publication and multicast it to all interested subscribers. For Reliable QoS, which is proposed in chapter 4,

the RN receives the acknowledgment from all subscribers and then sends one ACK to the corresponding publisher to release the next publication. Both the discovery and data dissemination phases are depicted in Figure 39.

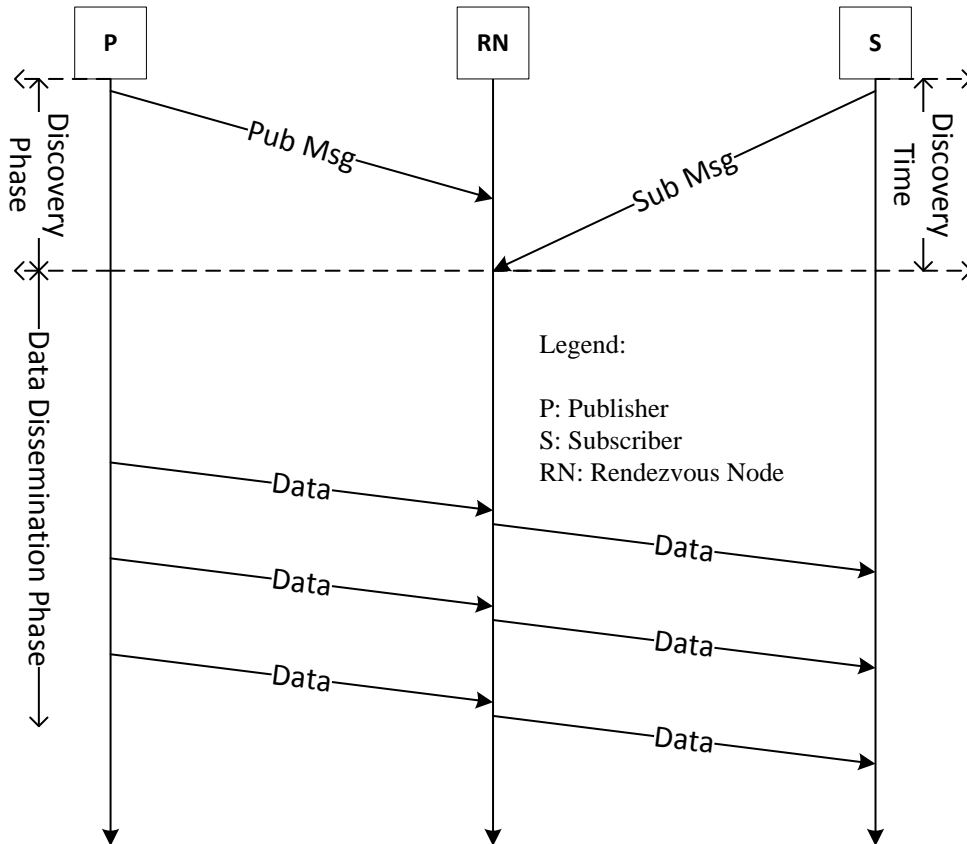


Figure 39 The sequence diagram of the discovery and data dissemination phases of DefTDDS

6.1.2 Broker-Less TinyDDS

In this method the RN node is totally eliminated, where the whole middleware functionality is distributed over the publisher and subscriber nodes. As shown in Figure 40, in the discovery phase, the subscriber broadcasts its subscription to all publishers. Thereafter, every publisher will do the matching process to decide whether it would send its data to this subscriber or not based on the matching process result. In case there is a match, the data dissemination phase starts, where publishers who have the requested/interested topic

start sending data to the interested subscriber. By eliminating the RN node in this approach, the single point of failure and bottleneck problems are solved. Moreover, this method distributes the network load over the network's nodes in more efficient way than the default method. However, since it is a flooding based approach, the number of control messages, i.e. subscription messages, highly increases. Consequently, high message dropping occurs and discovery time would be quite high compared to DefTDDS.

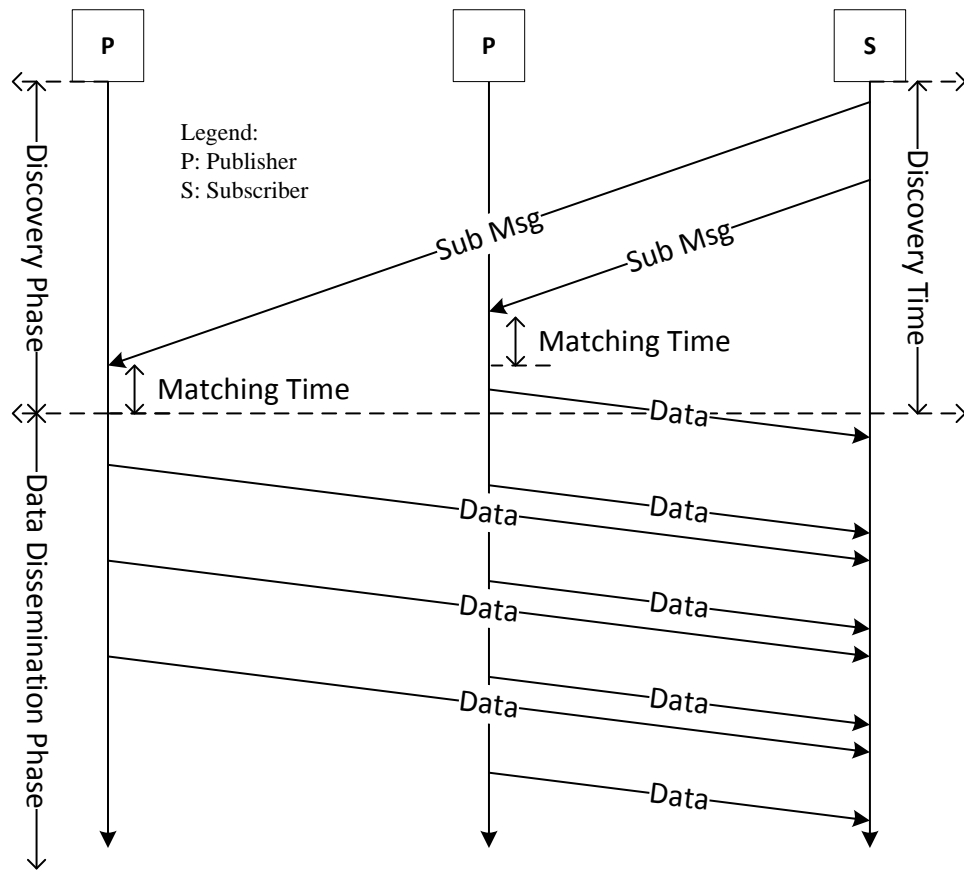


Figure 40 The sequence diagram of the discovery and data dissemination phases of BLTDDS

6.1.3 Hybrid TinyDDS

To minimize the discovery overhead of the BLTDDS while mitigating the negative effect of the bottleneck problem in DefTDDS, we propose a hybrid method that uses the RN in

the discovery phase and then totally eliminate it in the data dissemination phase. In this method, the subscribers list (subscribers data base) will be distributed over the all publishers; i.e. each publisher will maintain its all interested subscribers. Accordingly, in the discovery phase, the RN function is only to forward the subscription messages to the matched publishers, in this work we call the messages that is sent from the RN to the corresponding publishers a notification message. These messages are counted as overhead messages in the discovery process and calculated in our performance evaluation of this method. The sequence diagram of this method is depicted in Figure 41, in this figure we can see that after the discovery phase the data is directly sent to the interested subscribers.

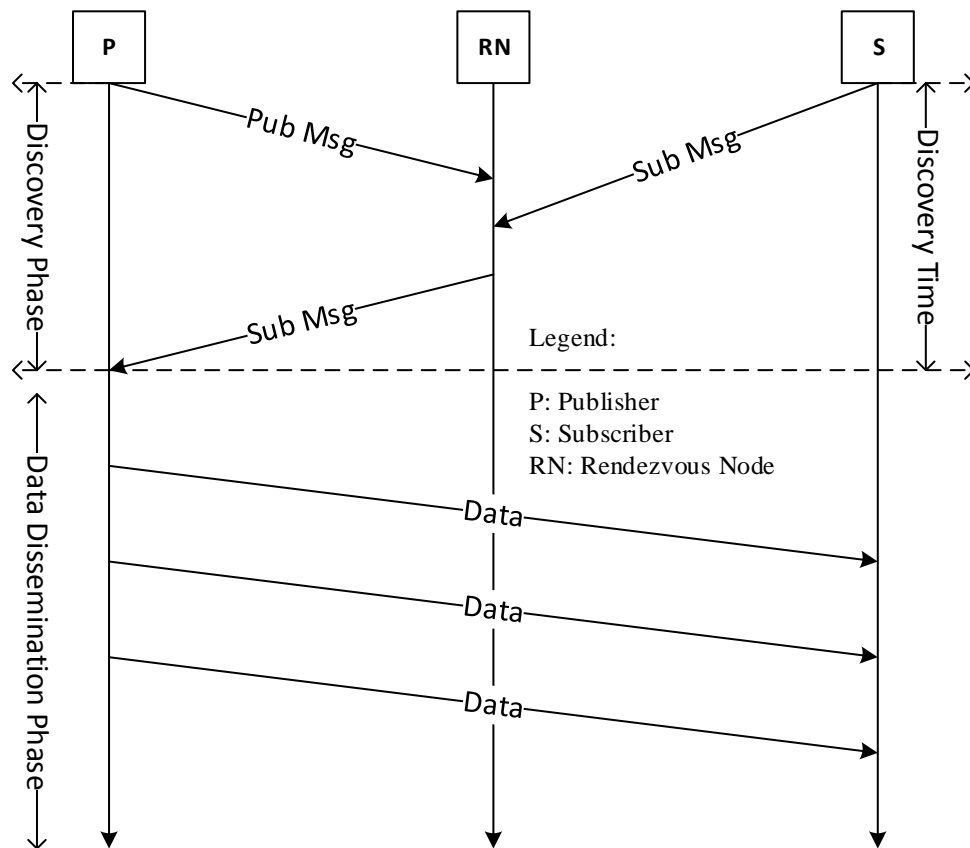


Figure 41 The sequence diagram of the discovery and data dissemination phases of HyTDDS

6.2 Performance Evaluation

6.2.1 Test Scenario and simulation setup

Since we are studying the sensor/actuator networks, this type of networks includes many of both types: publishers and subscribers. In the common scenario used in WSN applications, each set of sensors has one actuator that does the response for sensor readings, for example in heat monitoring systems, the sensors monitor the heat of the system and sends the readings to the cooling valve (actuator); and accordingly the actuator opens or closes the cooling valve. In this test we use small scale scenario, in terms of number of publishers, that tries to simulate the real applications of WSN. To the best of our knowledge this is the first time TinyDDS is tested under multiple subscribers and topics. The only difference between this scenario and the one used in reliability chapter is that while the reliability scenario uses the traditional WSN, this scenario adds more subscribers and topics to the network while the number of nodes remains the same. Figure 42 depicts the tested scenario that is used in our simulations in this chapter; it includes five publishers, and three subscribers that simulate two actuators and one common base station. The

actuators subscribe to different topics while the common base station subscribes to all the network topics, which are three topics.

This scenario was inspired by recent pub/sub middleware research called PS-QUASAR [19] which was described in the literature review chapter. The simulation setup parameters are shown in Table 10.

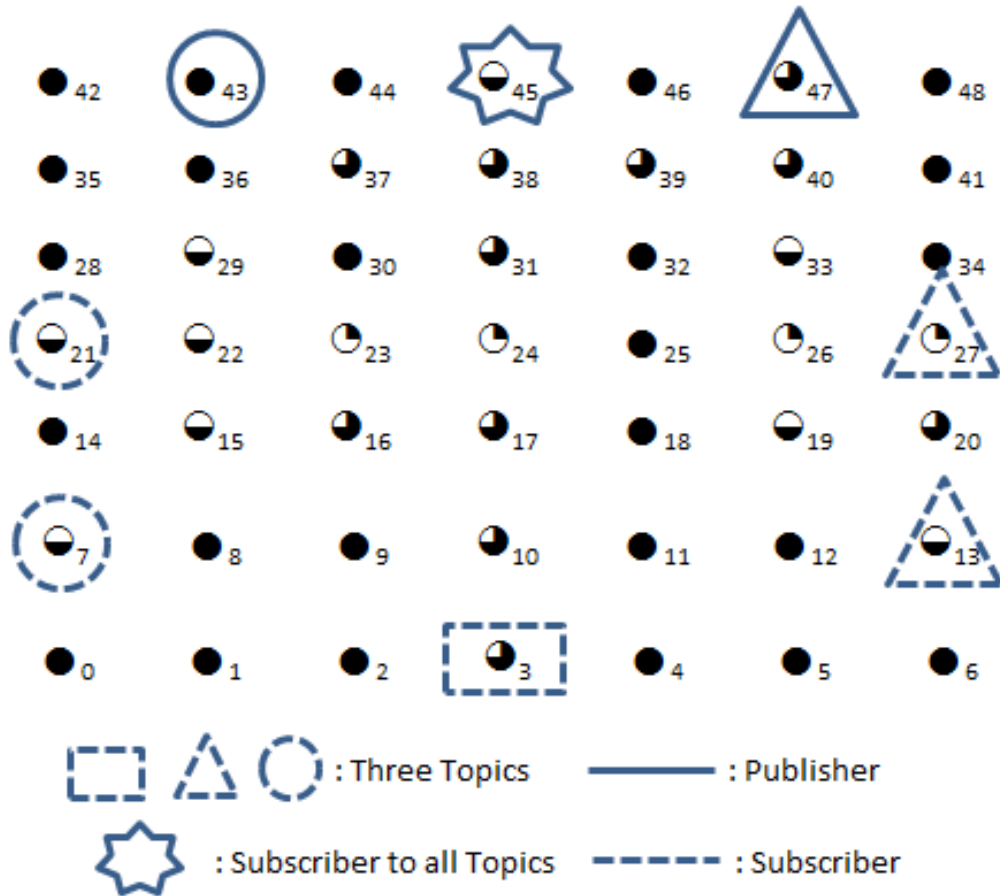


Figure 42 The tested scenario with 5 publishers, 3 subscribers and 3 topics

Table 10 Simulation setup

Parameter	Value
Topology	Squared grid
Area	100 X 100 Meter ²
Number of Nodes	49
Simulation time	1000 seconds
Radio model	Chipcon CC2420 [123]
Mote platform	micaZ
Data rates	30, 15, 10, 8, 5 Msg\Minute
Number of publishers	5
Number of subscribers	3
Number of topics	3
Message size	20 bytes
Runs per results' data point	10

6.2.2 Performance Metrics

Discovery Time

It is the time from the moment a subscriber join the network and send its subscription message up to the time the joined subscriber is recognized by the middleware, i.e. inserted into the middleware data base. In our test, since we have three subscribers the maximum discovery time is taken. It is important to notice that we assume that the middleware already knows all the publishers in the network.

- In Default TinyDDS (DefTDDS): it is the time from the moment the subscriber sends its subscription message until this message is recognized by the Rendezvous Node, since this node does the middleware core functions, e.g. matching and routing subscriptions and publications.

- In Broker-Less TinyDDS (BLTDDS): since there is no RN, the subscriber broadcast its subscription throughout the entire network. Thus, the discover time is calculated as the time from the moment the subscriber broadcast the subscription message until this message is recognized by the last publisher who has the interested subject.
- In Hybrid TinyDDS (HyTDDS): since the RN still exist in HyTDDS the subscription messages is headed to the RN, and then the RN will distribute the subscription message to all the corresponding publishers. Thus, the discovery time is composed of two main hops: (1) from the subscriber to the RN; (2) from the RN to the publisher. Since there is more than one publisher, we take the maximum discovery time.

Discovery messages

The discovery messages are messages used in the discovery process. Specifically, the subscription messages in case of DefTDDS and BLTDDS, also the subscription and notification messages in case of HyTDDS. Notice that in case of BLTDDS, each node that will rebroadcast the subscription message is also counted.

Packet Delivery Ratio (PDR)

The PDR is calculated by dividing the total number of successfully received messages at the subscriber side by the total sent messages from the publisher side. In case of multiple subscribers and topics, as in our test, we consider messages sent by all publishers as well as all successfully received messages by all subscribers. If PDR is less than one, that means there is packet dropping in the system.

End-to-End Delay (EED)

The EED is measured from the moment of sending/publishing data on a publisher side until it is successfully received on a subscriber side. The delay is calculated for all successfully received messages by all subscribers and then the average is taken.

Energy consumption

The energy consumption is calculated by taking the summation of energy consumption of all the network nodes in milli-Joule. Energy consumption also gives a relative indication of the network life time. As discussed in the OEM chapter, the radio and MCU are the only components that will be considered in our evaluation.

6.2.3 Results and Analysis

One of the most important performance metrics of any pub/sub system is the discovery time, specifically for real time systems, and discovery overhead, represented by the number of discovery messages, i.e. subscriptions and notifications. In Figure 43, the discovery overhead of the three methods is depicted. The discovery messages seem to have no effect by the data rate, because we use the default approach in sending the subscriptions messages in TinyDDS, in which this approach sends a constant number of messages, ten messages, to assure the receiving of the subscription messages. Thus, mostly it will send the same number regardless of the data rate or IPI. The BLTDDS shows the highest overhead because it uses a flooding algorithm to distribute the subscription messages, while DefTDDS has the lowest overhead because it does not use broadcast subscription messages or notification messages, i.e. from the RN to the publishers in case of HyTDDS. The

HyTDDS method is the only one that has notification messages; therefore, its discovery overhead is slightly more than the DefTDDS.

In Figure 44, the discovery time is depicted, and as shown in the figure we can see that the DefTDDS is still the best in terms of discovery time, and that is because it has only one overlay hop, i.e. from the subscriber to the RN, to be discovered by the middleware core system. Whereas in HyTDDS, it needs two overlay hops, i.e. from the subscriber to the RN and then from the RN to the corresponding publishers. That means, as the number of the corresponding publishers increases the system incurs more delay; thus, HyTDDS is the worst in terms of discovery delay. Although BLTDDS is the worst in terms of overhead, it has small delay the broadcasting is very fast to reach all the corresponding publishers. Intuitively, the discovery time should be less in case of low data rate, and vice versa, as shown in case of DefTDDS and BLTDDS. However, this is not the case in HyTDDS, where from the error bars, which represents the standard deviation of the repeated runs, it seems not stable specifically in cases eight and ten IPI. That might be, as we mentioned earlier, because it sends two overlay hops which increases the probability of collisions and thus packet loss. At the end of the discover process analysis we can conclude that the default TinyDDS (DefTDDS) is still the best choice for real-time WSN networks in terms of discovery delay and overhead.

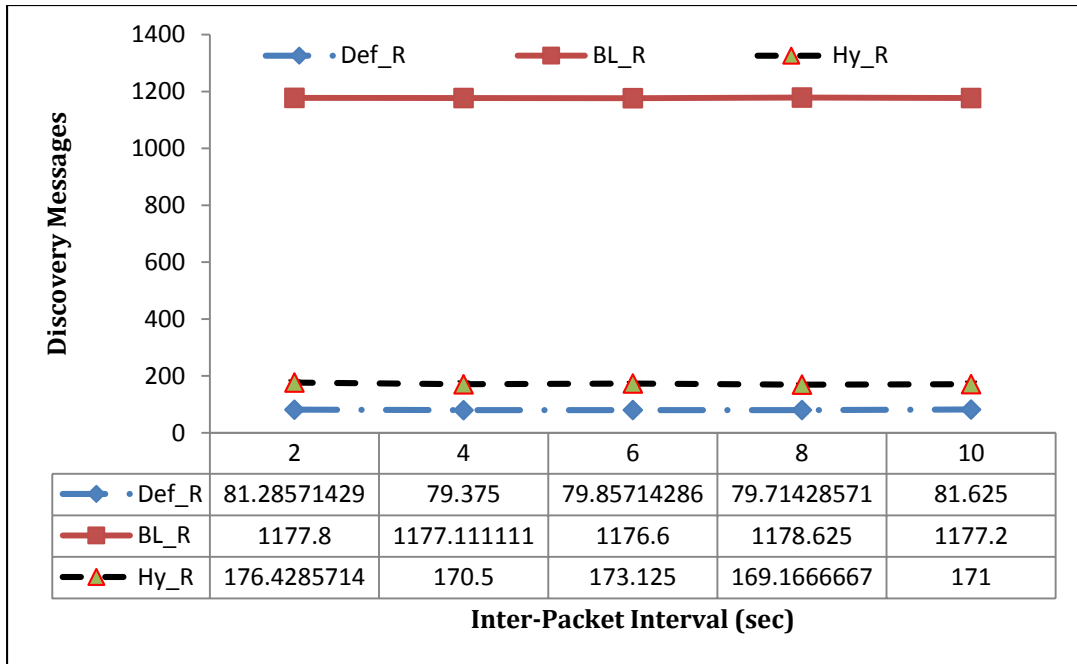


Figure 43 The number of discovery process messages vs. IPI

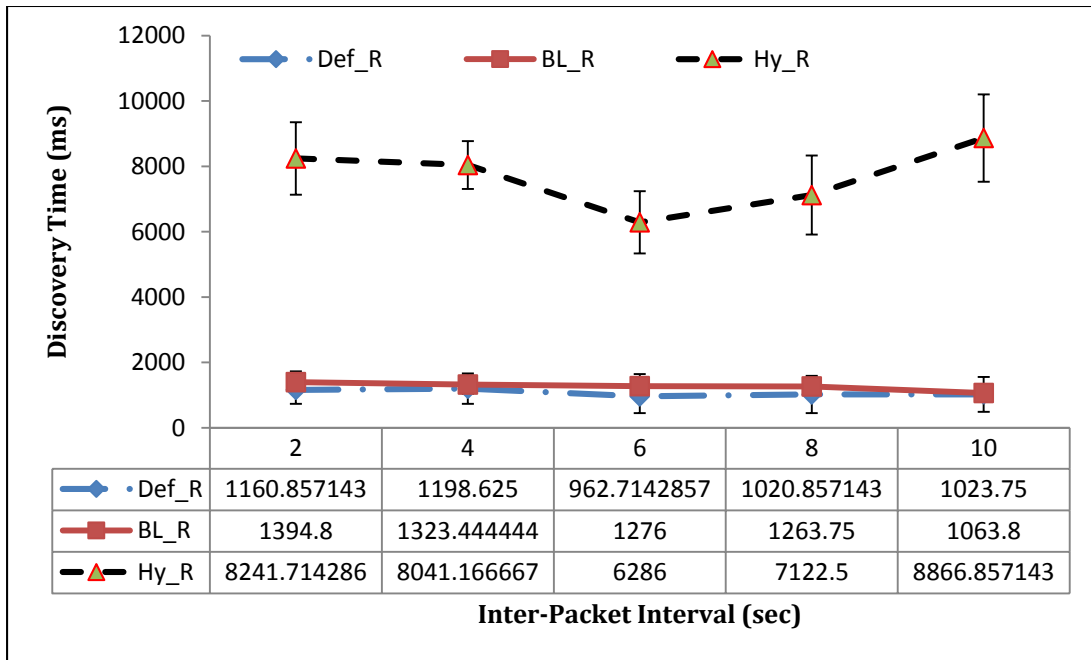


Figure 44 The average discovery time of new subscriber

Intuitively, the bottleneck problem of the DefTDDS will make it the worst case in data dissemination. However, in low data rate it performs as effective as BLTDDS and

HyTDDS. In Figure 45, it is obvious that the BLTDDS and HyTDDS further improves the Reliable TinyDDS, in which they completely eliminate the packet dropping in case of heavy data load, i.e. IPI one second. This is due to the load distribution over the network nodes rather than enforcing the data to go through a single point, i.e. the RN. The same trend can be shown in case of using best effort QoS, as shown in Figure 46. Since the BLTDDS and HyTDDS methods use the same method in data dissemination phase, results are almost the same in both cases, i.e. reliable and best effort scenarios. In general, the PDR is very low in case of best effort scenario compared to the results of RTDDS tests. That is because of using 3 subscribers instead of one in RTDDS test. Unlike increasing the number of the publishers, increasing the number of subscribers has a significant effect on the network performance, since each subscriber needs to receive the data from most of the publishers, especially if one topic is used in the network.

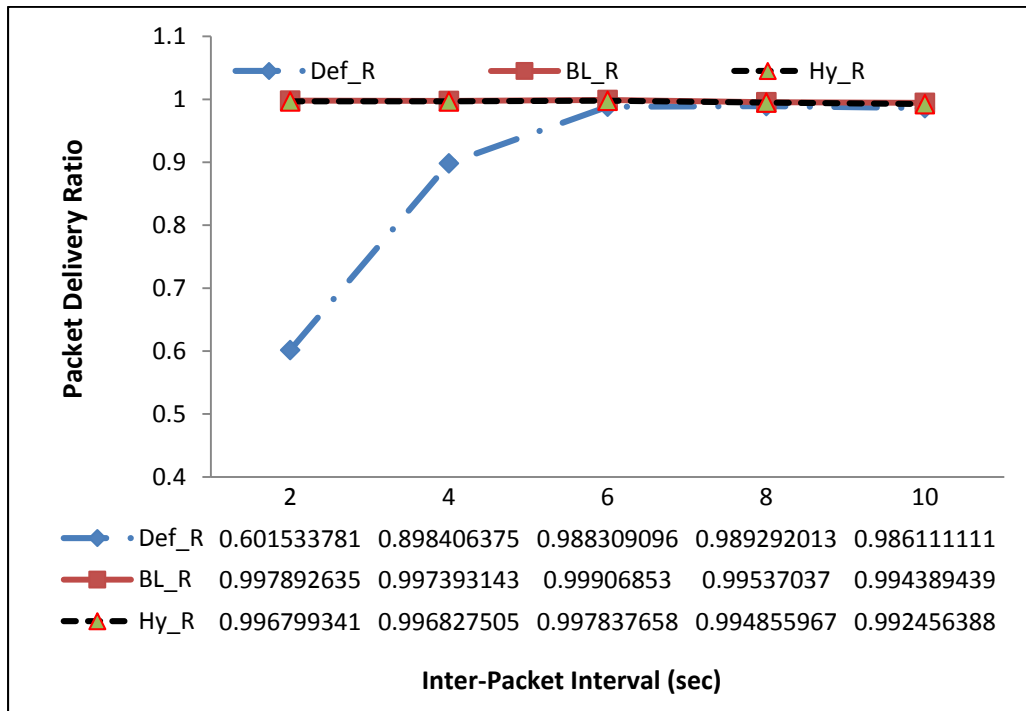


Figure 45 Packet delivery ratio of the reliable scenario

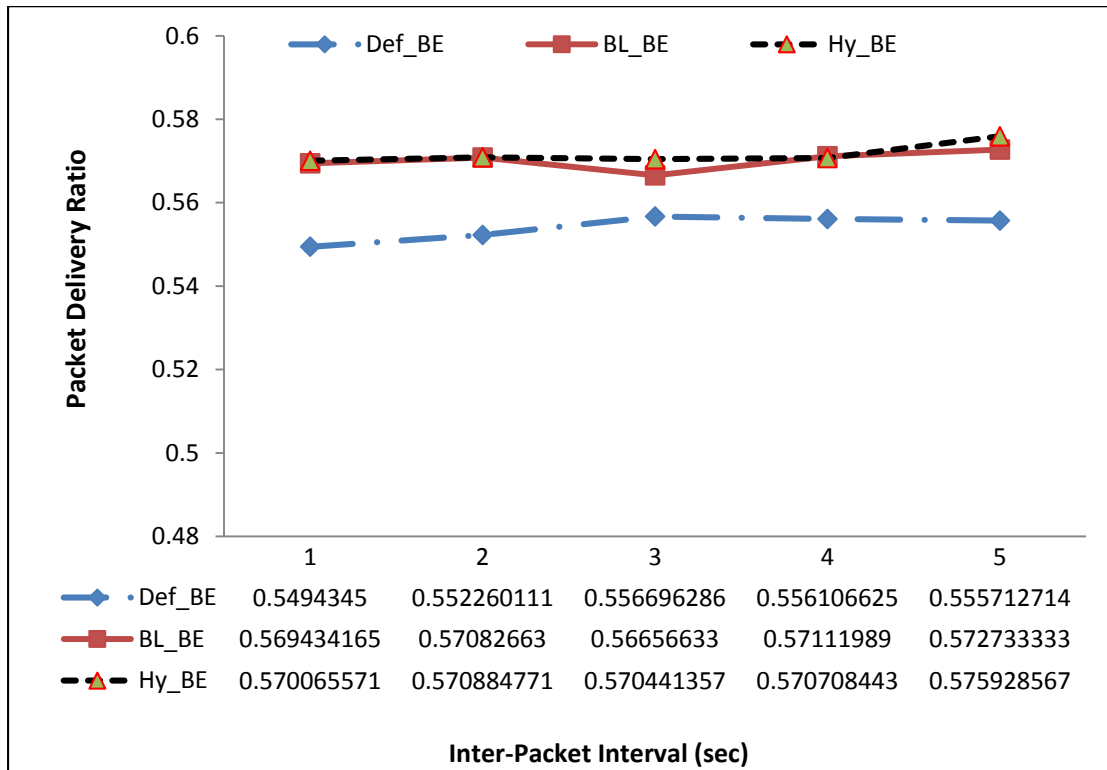


Figure 46 Packet delivery ratio of the best-effort scenario

The results of the PDR can be reflected in the End-to-End Delay (EED) results. Both Figure 47 and Figure 48 are depicting the EED of the scenarios reliable and best effort respectively. The improvement of the BLTDDS and HyTDDS is very clear, specifically in the heavy load cases. The EED of the BLTDDS and HyTDDS is in the range of hundreds of milliseconds. In reliable scenario, the delay decreases as the network load decreases, i.e. IPI increases. In contrast, in the best effort scenario, we can observe that the effect of the data load on the EED is not significant, that may be due to the fact that the publishers are sending at the same time, exactly at the end of each interval; which makes the channel contention and thus packet dropping almost the same in different IPIs. For this reason, we used Interference Free Scheduling (IFS) in the reliability test in chapter 5.

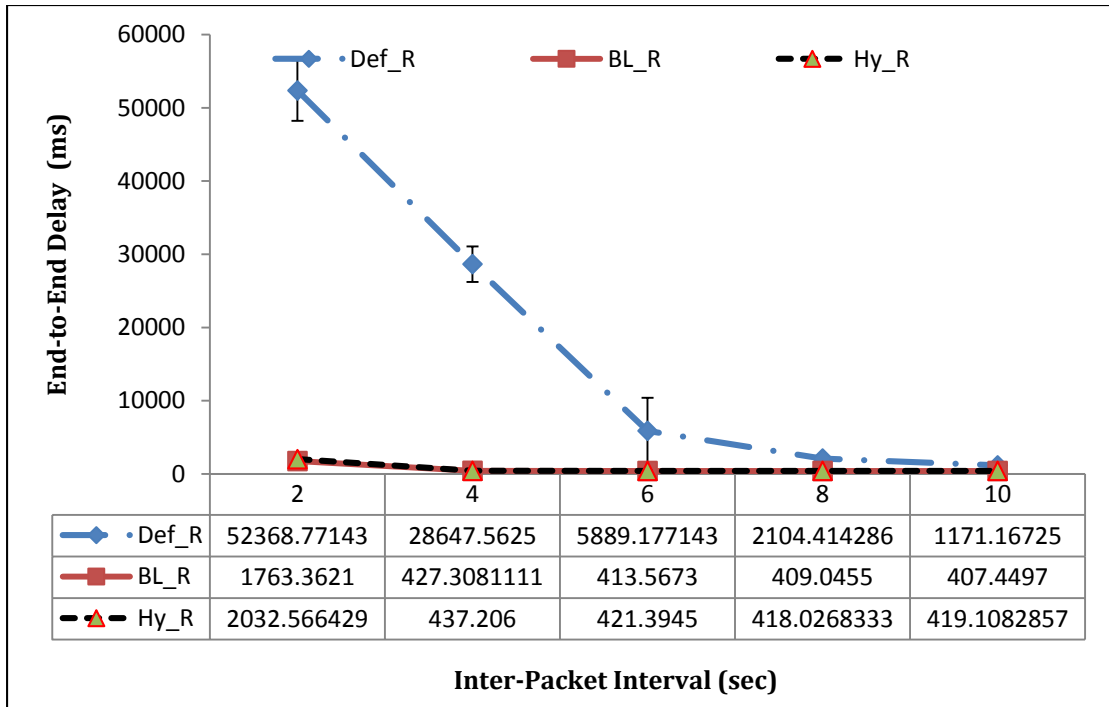


Figure 47 End-to-End scenario of the reliable scenario

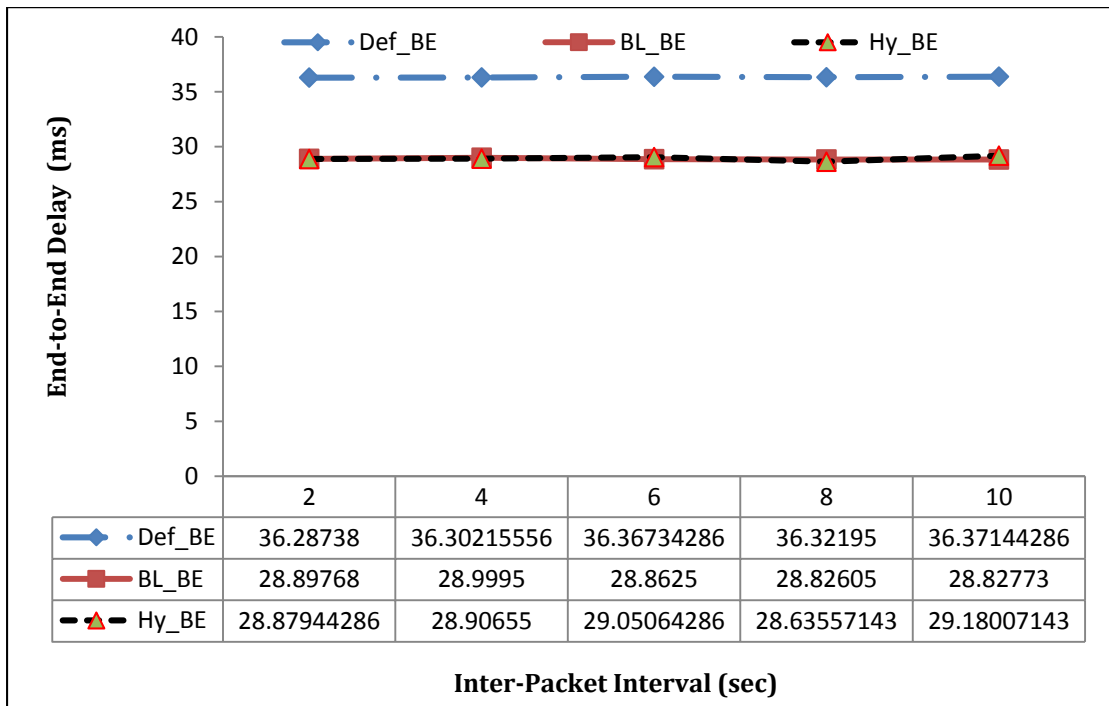


Figure 48 End-to-End delay of the best-effort scenario

Energy Consumption

In this section we discuss and analyze the energy consumption results of the three methods, DefTDDS, BLTDDS and HyTDDS. The used energy model parameters are shown in Table 11.

Table 11 MicaZ Energy model parameters

MCU		Radio	
Mode	Current	Mode	Current
Active	8 mA	Receiving	19.7 mA
Idle	4 mA	Transmitting	17.4 uA
Sleep	9 uA	Sleep	1 uA

The energy consumption was tested under the reliable and best effort scenarios and computed as total energy consumption of the radio and MCU components, as shown in Figure 49 and Figure 50. The results show that the BLTDDS and HyTDDS outperform the DefTDDS by nearly one third, in the reliable scenario. That is because in case of DefTDDS, the network is instable, as shown from the PDR results in Figure 45, specifically with high work load, e.g. IPI equals 2 and 4. This increases the number of retransmissions, which in turn increases the energy consumption as well. In contrast, in case of best effort scenario, as shown in Figure 50, the three methods nearly seem to have the same energy consumption and that is because the total send and receive messages are almost the same, except that in case of DefTDDS the messages may take longer paths due to the existence of the broker or RN. The BLTDDS and HyTDDS results are almost the same except that there is a slight increase in case of BLTDDS in both scenarios, i.e. reliable and best effort, and that is

because the flooding messages in the discovery phase. Therefore, HyTDDS seem to be the more efficient protocol in terms of energy consumption, in both reliable and best effort.

This observation will be clearer from the next individual results of the three methods.

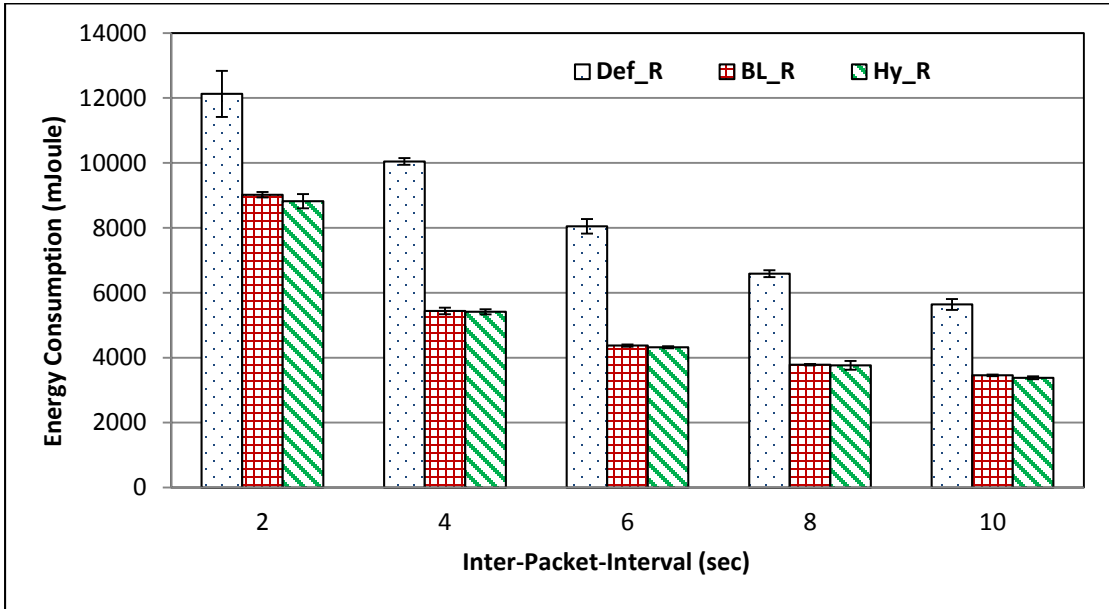


Figure 49 Total energy consumption in milli-Joule for reliable QoS

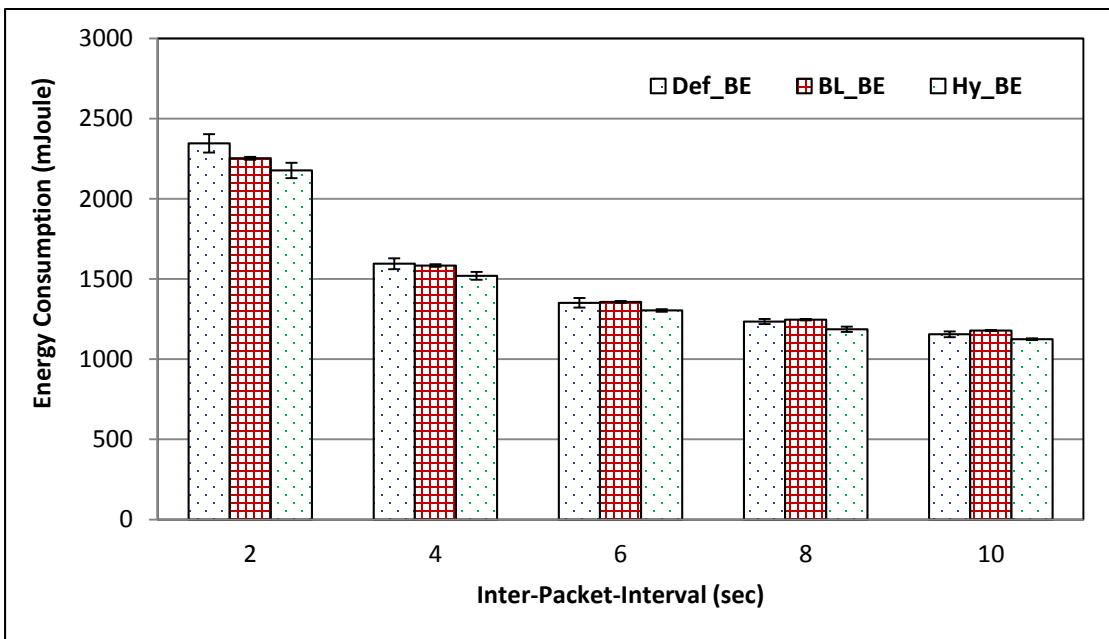


Figure 50 Total Energy Consumption in milli-Joule for Best Effort QoS

The previous results showed the total energy consumption, for both the Radio and MCU components. In this part, each method is evaluated separately in terms of energy consumption of both radio and MCU components, as shown in Figure 51 and Figure 52. The BLTDDS and HyTDDS results are nearly the same; therefore, the results tables were added to give the exact difference in their performance; specifically the effect of the flooding approach on energy consumption. In general, the MCU energy consumption is nearly one fourth the consumption of the radio, which is very high compared to the new sensor platforms, such as TelosB, iris and Zolertia, where the consumption of the MCU is almost neglected when compared with the radio consumption. That is because the MCU energy consumption in the new platforms is very small compared to MicaZ platform. For example, in case of TelosB, the MCU active mode current consumption is 1.8 milli-Amp whereas it is 8 in case of MicaZ; and the sleep mode current consumption of TelosB is 3 nano-Amp whereas it equals 9 micro-Joule in MicaZ. An important observation is that most of the MCU time is in the idle or sleep mode, that may be the reason for having almost the same MCU energy consumption while the network load increases.

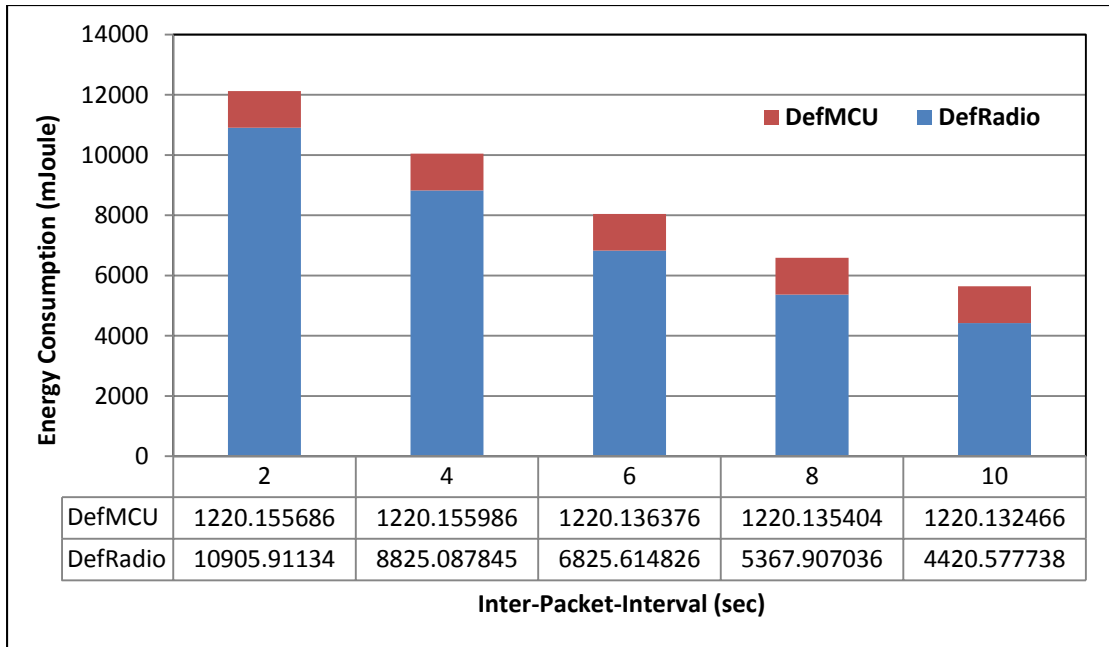


Figure 51 Radio and MCU energy consumption of DefTDDS with Reliable QoS

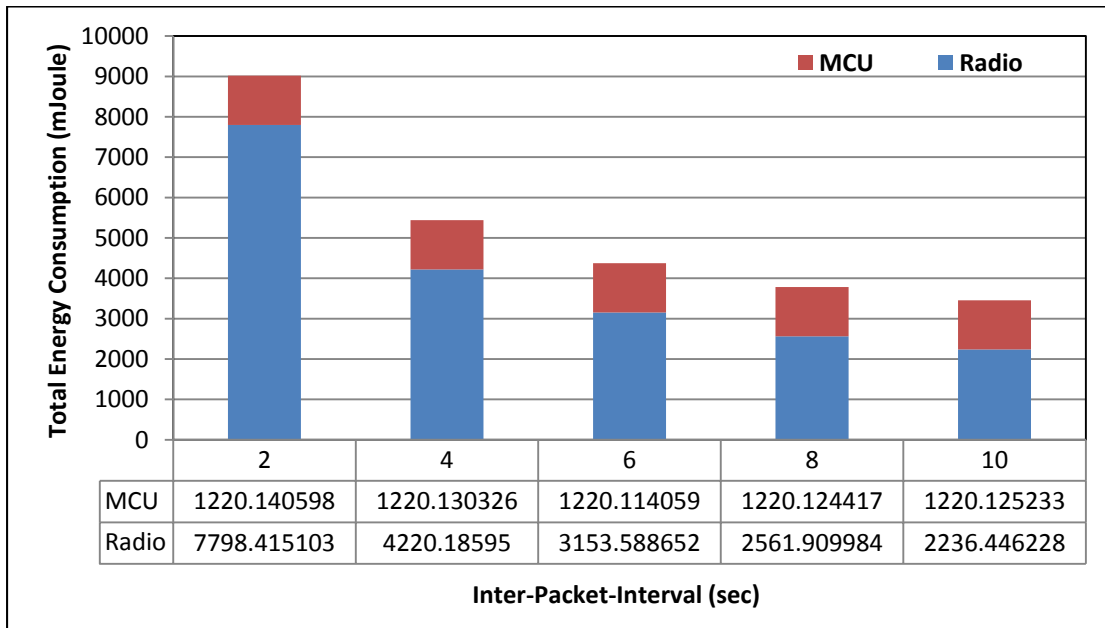


Figure 52 Radio and MCU energy consumption of BLTDDS with Reliable QoS

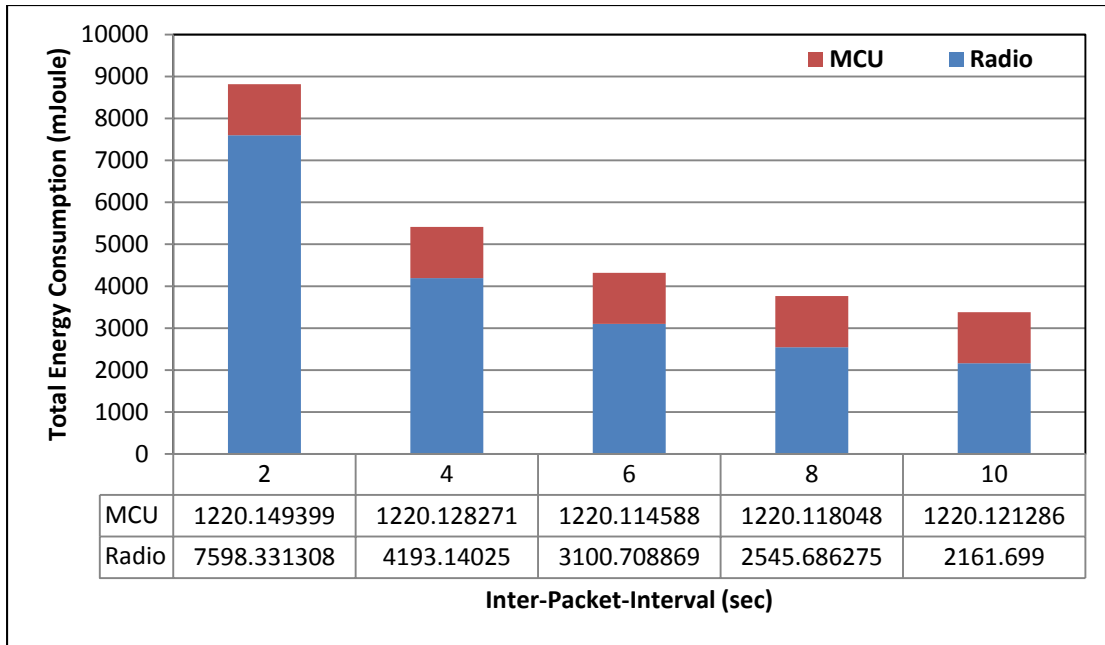






Figure 53 Radio and MCU energy consumption of HyTDDS with Reliable QoS

The main target of the following results is to explore the energy consumption distribution of the three methods over the network nodes. In each figure, we pinpoint every subscriber, publisher, relay, rendezvous nodes (brokers) and topic/data types. Table 12 describes the symbols used in the energy consumption distribution figures (Figure 54, Figure 55 and Figure 56). We selected the most heavy load network scenario to be depicted in this evaluation, which is the reliable with two seconds IPI. For the sake of highlighting the impact of the energy consumption distribution on the network life time, for each method we compute the energy consumption of each node in the network relative to the maximum energy consumption in the network. For example, if the maximum energy consumption was in node B, equals 500 milli-Joule (mJ), then node B energy consumption equals 100%; and for the remaining nodes in the network, the energy consumption for every node equals the energy consumption of the node divided by node B energy consumption, as shown in equation 6.2. Therefore, this evaluation does not show the total energy consumption of each method, since this metric has been covered in the previous results. Also, each node in

the network is represented by a battery shape with accuracy of 5%, i.e. any change in the energy consumption in the range of 5% will be reflected in the battery (node).

$$EC_{relative} = \frac{EC_{absolute}}{EC_{Max}} \quad (6.2)$$

Table 12 Energy consumption figures' symbols

Key	Meaning
	Subscriber
	Rendezvous Node
	Publisher
	Topics/data types

In Figure 54, the energy consumption distribution of the DefTDDS method is depicted. As shown in the figure, only one node is considered as a dead node, which is the one having the maximum energy consumption. Thus, if we define the network life time as the time starting from the deployment moment until the first node is dead, the remaining energy may be considered as a wasted energy. The remaining energy is depicted in the figure by the remaining energy in the node batteries. Thereby, from the figure we can see how much is the wasted energy in DefTDDS method, which is the worst case in this study. It is very obvious that we really need an energy aware mechanism that can distribute the load of the network based on the network energy consumption distribution, which will be introduced in the next chapter. As discussed earlier, since we have three topics in this network then the default TinyDDS middleware dedicates three RN nodes, one for each topic. These nodes are shown in the figure, which can be exactly pinpointed using Table 12, and we can see that the most exhausted nodes are those which transfer the data from RN nodes into the

base stations; this observation is because the whole data that comes from the all publishers are transferred throughout these nodes. The middle RN node, the RN node of the circle topic, is the only one from the RN that still away from about to die, that is because this RN has only one publisher; whereas the other two RN nodes have two publishers each. About 23 from 49 nodes are still having full batteries while the network life time is over because one of the RN is already dead and the other one is about to die; consequently, four publishers are considered totally disconnected from the network. Notice that we are discussing the effect of the middleware layer independent from the underlying network protocols. Therefore, we do not discuss the effect of the underlying routing protocol on the energy consumption distribution; however, it has a significant effect on the performance regarding energy consumption.

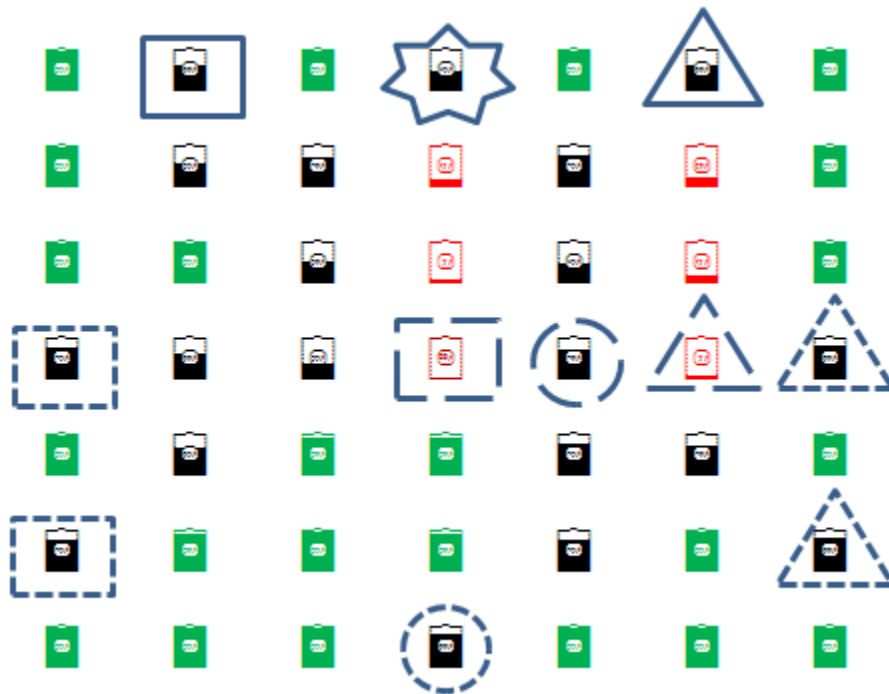


Figure 54 DefTDDS energy consumption distribution over the network nodes

The other two methods, BLTDDS and HyTDDS, almost have no RN nodes; that means they are totally dependent in the underlying routing protocol in distributing the energy consumption over the network nodes. Figure 55 and Figure 56 depict the energy consumption of BLTDDS and HyTDDS, respectively. The two results are nearly the same, since they have the same method in data dissemination phase which is the dominant phase in any pub/sub middleware. The slight difference is reflecting the difference between the two methods in the discovery phase, where BLTDDS uses a flooding algorithm and HyTDDS uses the RN nodes, as shown in Figure 56. Since these methods totally eliminate the RN nodes in data dissemination, i.e. the bottleneck problem, the data dissemination distribution over the network nodes is much better than in DefTDDS; which is reflected into the energy consumption distribution. However, since we do not use an energy aware routing protocol in the underlying layer, the network still have a lot of remaining energy (wasted energy), as shown in the figure, 18 out of 49 nodes are still having full batteries. Note also that the middle base station has almost exhausted its energy, because it subscribes to the all topics in the network, e.g. data base server.

This chapter leads us to the next chapter, where we introduce the solution for the energy consumption distribution problem in DefTDDS. Although the other two methods have significantly improved the function of the default TinyDDS, specifically in the data dissemination phase, still they are not an energy aware methods and are totally dependent on the underlying layers in the energy consumption distribution.

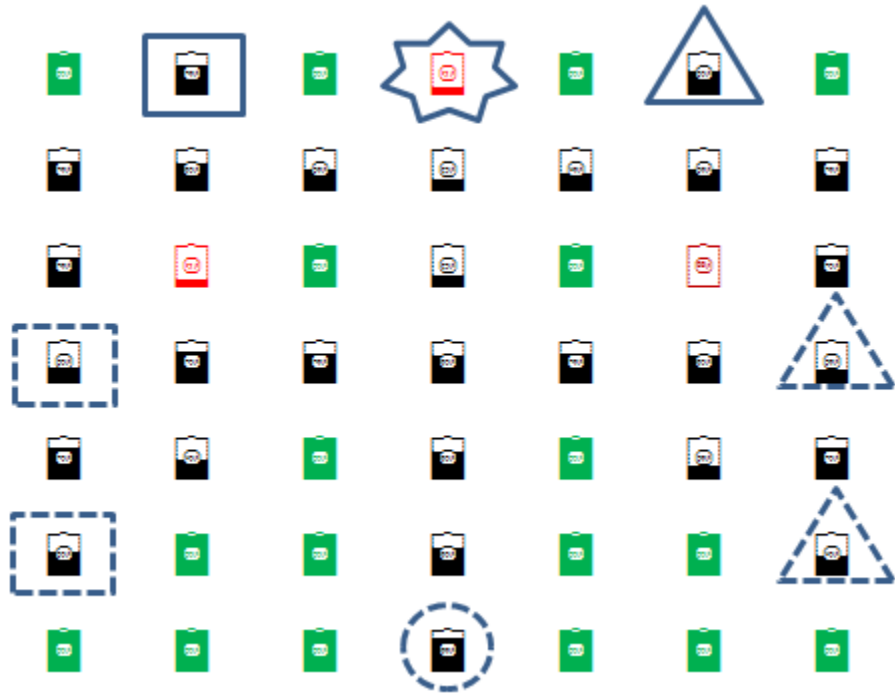


Figure 55 BLTDDS energy consumption distribution over the network nodes

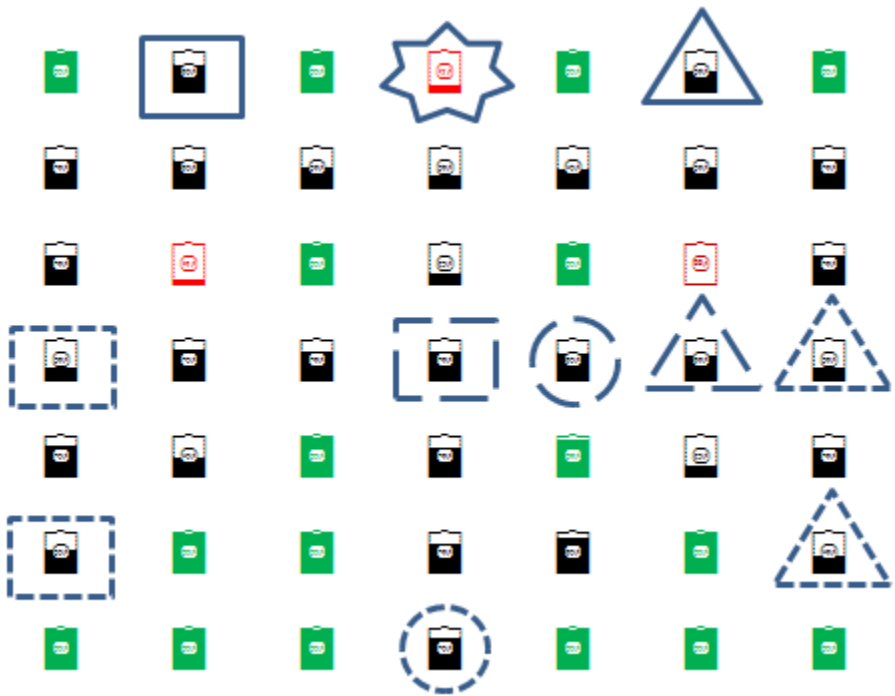


Figure 56 HyTDDS energy consumption distribution over the network nodes

CHAPTER 7

EATDDS

Energy is a very critical resource for sensor-based networks. Most of the sensors/actuators use AA size batteries, and changing these batteries is costly and in some cases very difficult, e.g. hazard or harsh regions monitoring. Therefore, minimizing energy consumption and developing energy aware protocols in WSN is currently a hot area of research. In this chapter, we continue TinyDDS enhancements by introducing the final enhancement, in which a publish/subscribe energy aware protocol based on DDS is presented and evaluated.

7.1 EATDDS Description

In this protocol we assume that the node location is known for all network nodes, e.g. using GPS devices, or localization protocols. As in our scenario, a grid topology is used which is the tested topology in the TinyDDS implementation test. EATDDS uses the location of the nodes to minimize the distances between the publishers and interested subscribers, thus minimizing the energy consumption. Since the energy consumption is directly proportional to the square distance between the sender and receiver [128]. The OEM that is described in chapter 5 is used in this work to monitor the energy consumption of the network nodes. Each node will monitor its energy level and based on the common round used by all the nodes, it will send its information periodically to the cluster RN node.

The EATDDS algorithm is inspired by the LEACH-C protocol [128], where our 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 57 shows how our network can be clustered into three main clusters, each cluster represents a distinct topic. In EATDDS algorithm each RN will be 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 new RN node its cluster. The new RN node will be selected from the cluster nodes as the one having the maximum remaining energy.

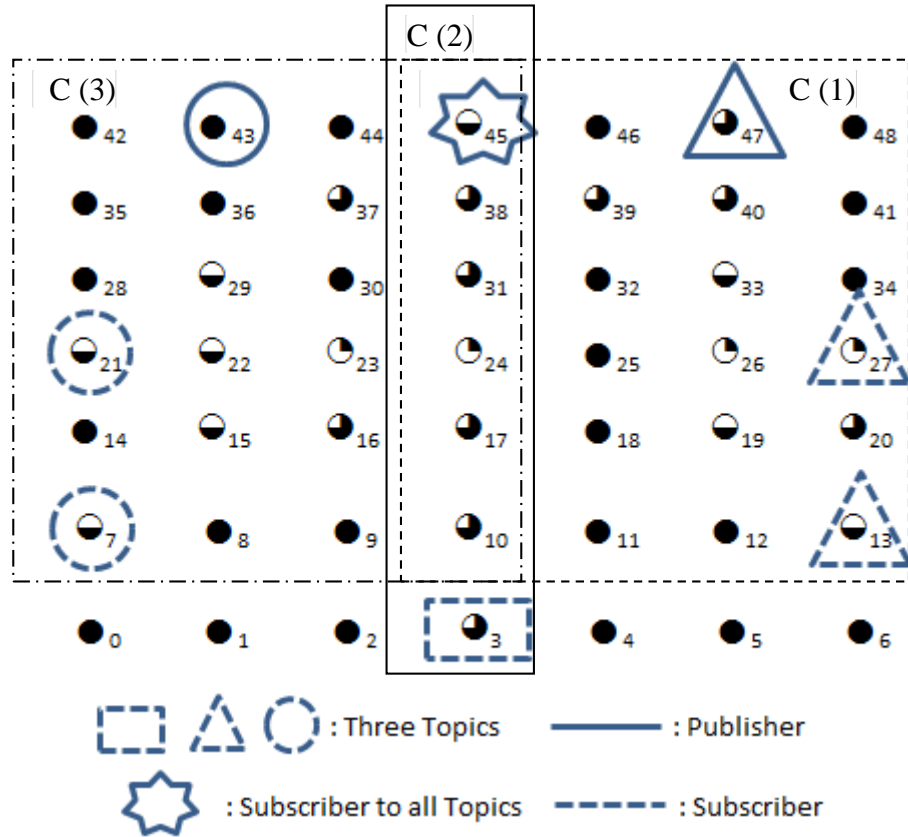


Figure 57 Cluster formation of EATDDS

Since all the nodes have registered the topic/data in the deployment phase, all nodes 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 exist, every node will send the energy updates to the all RN nodes in the network.

7.2 Performance Evaluation

In this section, EATDDS is extensively evaluated and tested under different network loads, represented by IPI. 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.

7.2.1 Experiment setup

The simulation set up and network topology is the same as in the OEM chapter, the topology can be shown in Figure 57. As mentioned above, the only difference is the unlimited simulation time, whereas in OEM simulations it was 1000 seconds, and in RTDDS it was 500 seconds, so it gradually increases. The new and most important parameter in this simulation is the initial energy; where all the network nodes will start with an initial energy, and once this energy is dissipated the node is considered dead. We select the initial energy to be one joule, as in LEACH-C paper [128]. Moreover, the data rate is constant, that means all the protocols are subjected to the same workload, which makes the comparison more fair. 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.

7.2.2 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)

This metric reflects the good distribution of energy consumption on the network nodes. Therefore, a large amount of wasted energy reflected 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.

7.2.3 Results and analysis

The total energy consumption in Figure 58, and wasted energy in Figure 59 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 the worst case since it has the most wasted energy while the network is over. Likewise, it has the largest total energy consumption, that means less work has been done in this protocol. Both the broker-less and hybrid protocols appears to be the most effective, and thus have the longest network time. EATDDS protocol is getting better with the work load decreasing, that is obvious from the difference of the TEC that is increasing with IPI increases.

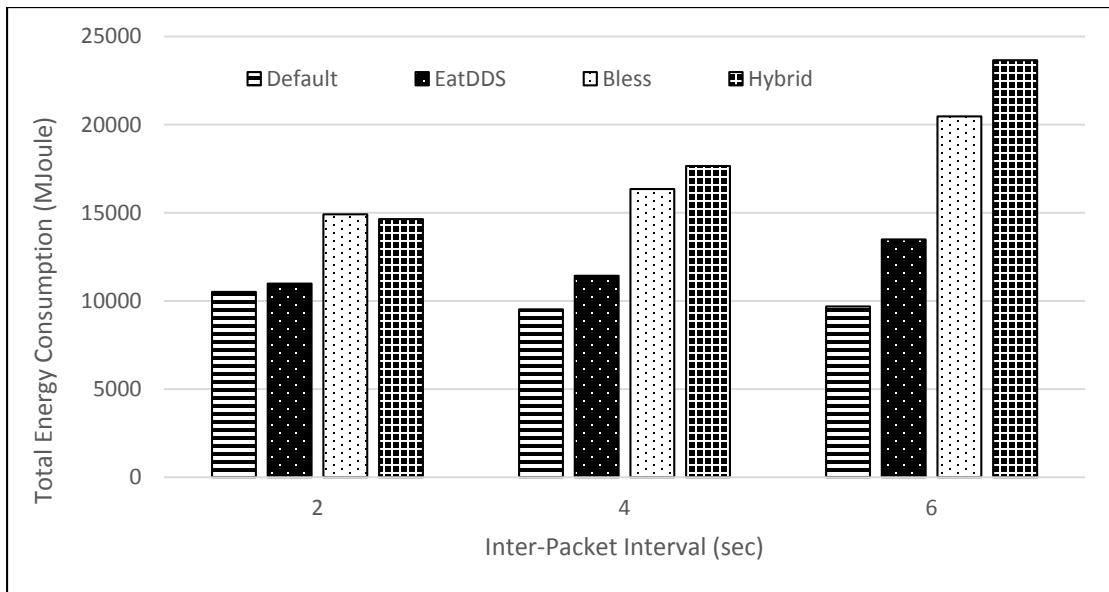


Figure 58 The network Total Energy Consumption

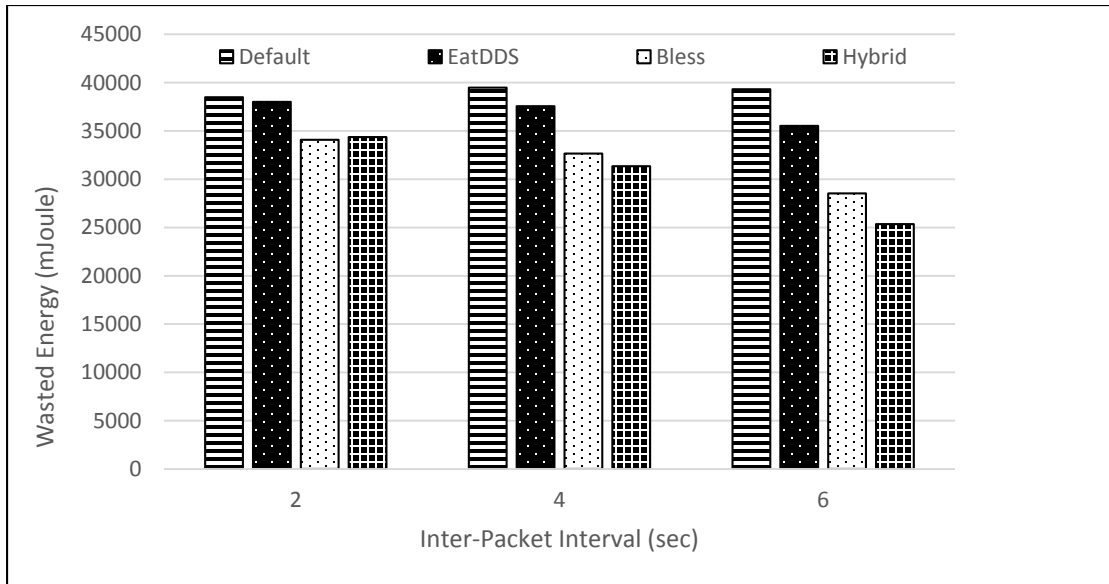


Figure 59 Remaining energy at the end of network life time

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 60, EATDDS protocol seems to be the best in case of less network load, while it appears the worst in case of the heavy network load. Due to the random selection of the RN node, EATDDS may behave inappropriate when subjected to heavy network load.

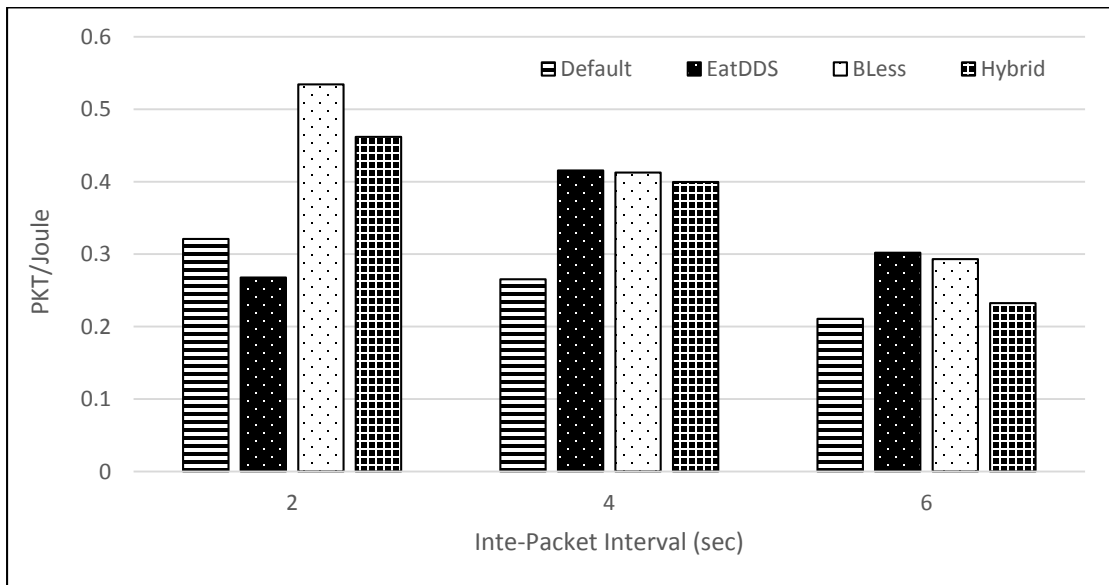


Figure 60 Packets per Joule vs. Inter-Packet Interval

In network life time, the broker-less and hybrid protocols nearly perform the same in different workloads, while EATDDS shows a significant improvement to the default TinyDDS. The broker-less and Hybrid, are almost the same technique except the process of the discovery phase, therefore, in long-term process they may converge to finally perform the same, as shown in the Figure 61.

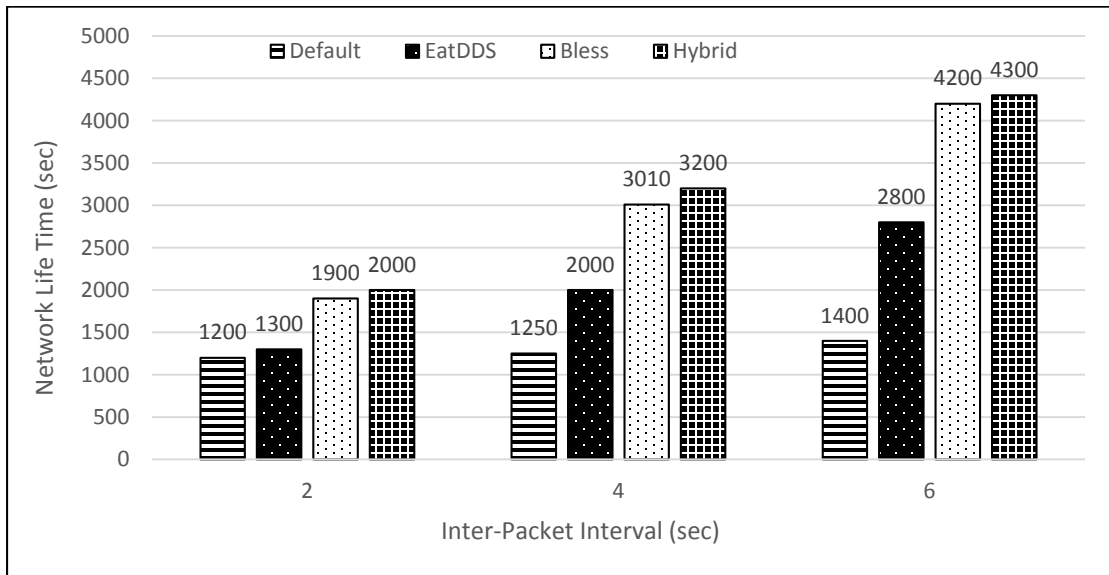


Figure 61 Network life time at the moment the first node dies

Prototyping

On one hand, the main advantage of TOSSIM simulator is that you are building a real implementation, that needs very small modifications to upload to real sensors and work normally. On the other hand, this is the same reason of the difficulty of this simulator, since doing any modification 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, as shown in Figure 62. In this experiment we tested the real energy consumption of TelosB platform when EATDDS is working on it

with different scenarios. These scenarios are: with central broker (RN) and with distributed brokers to the effect of the bottleneck on the real network performance. The TelosB motes are used without the low power listening 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 62 TelosB mote platform

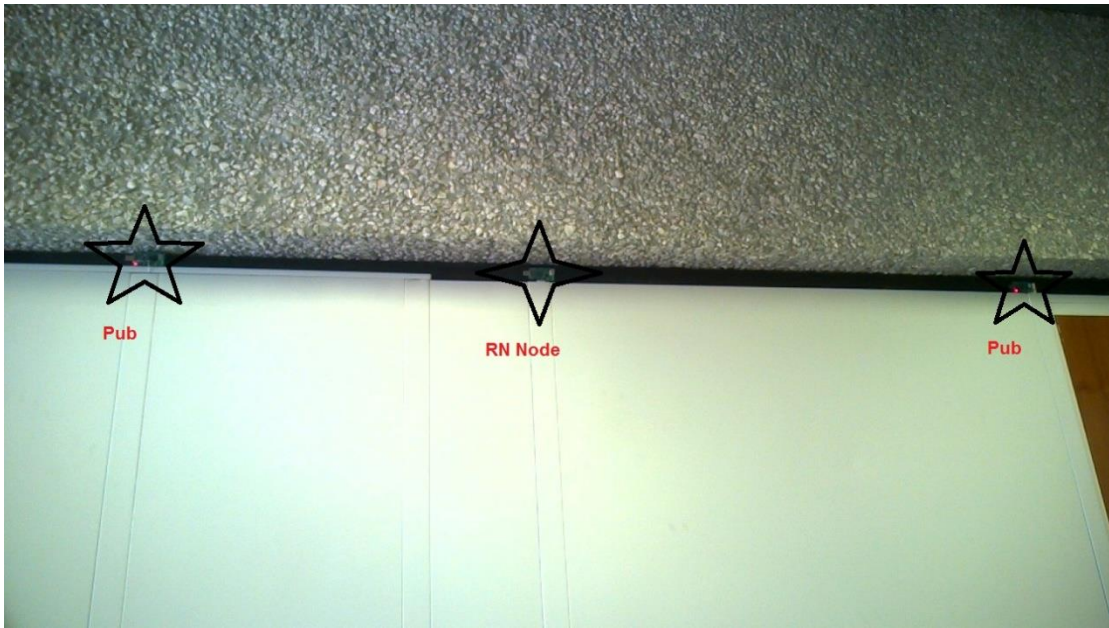


Figure 63 Experiment environment and testbed

The seven nodes are distributed indoor, i.e. inside the lab as depicted in Figure 63. There in each side, two publishers and one RN node; and the base station is placed directly on the USB port, as shown in Figure 64; however, also the base station were tested with new batteries to see the energy consumption in the base station nodes.



Figure 64 The Base Station attached to the PC USB port

The measurements the were taken are 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 13 shows the effect of the centralized approaches in real scenarios, where the distributed scenario relaxed the network more and thus minimizing the contention and consequently packet dropping and collisions. Furthermore, the standard deviation may reflect the instability of the centralized approach, since the all publishers of the network have to go through this central RN.

Table 13 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

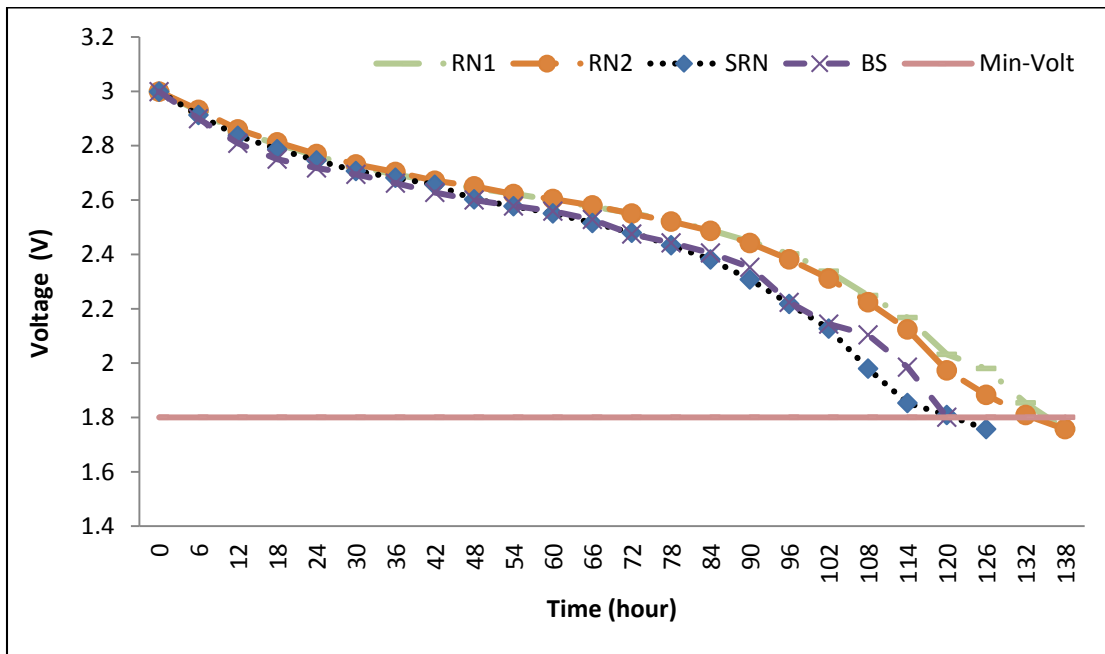


Figure 65 Network life time using 7 motes with AA energizer batteries

In Figure 65, the network life time can be estimated for all types of nodes, e.g. RN1, RN2, SRN (single RN node), BS (Base Station). The Minimum Volt is the minimum power of the TelosB motes to work properly [129]. As shown in the figure, 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 and that is intuitive 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 were all the time in the receive mode, that makes the difference between the all sensors quite small.

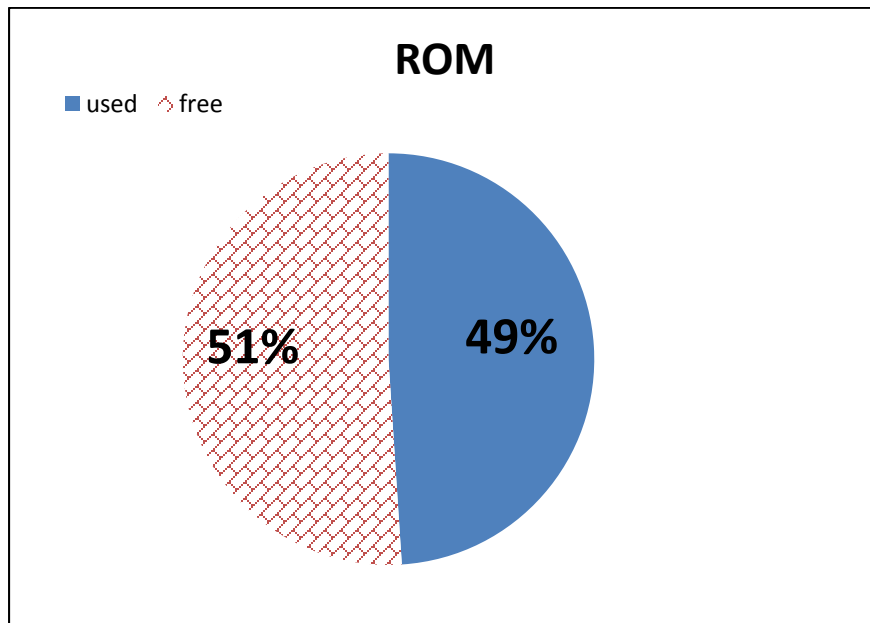


Figure 66 ROM occupied space after uploading EATDDS

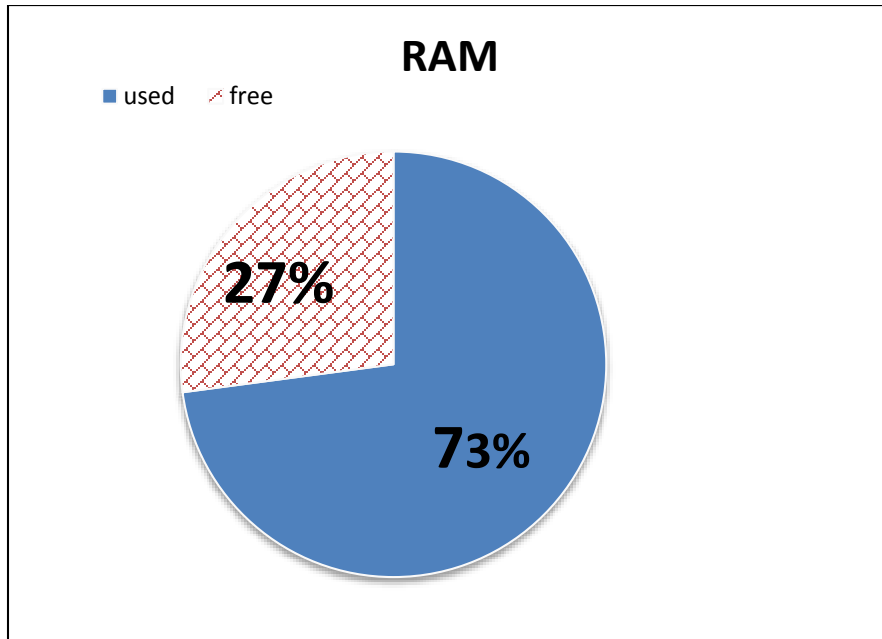


Figure 67 RAM occupied space after uploading EATDDS to TelosB mote

The memory is a very important measurement, specifically for the limited resources devices, e.g. sensors. It gives a clear evidence of the applicability of the developed technique. The memory measurements of EATDDS is shown in Figure 66 and Figure 67, for ROM and RAM respectively. The results show that the memory in both ROM and RAM still have free space around 51% and 72% 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.

CHAPTER 8

CONCLUSIONS AND FUTURE WORKS

In this work we reviewed the publish/subscribe interaction paradigm in the context of WSN/WSAN, we discussed its suitability, components, architectures, and variants. Also, we surveyed the state-of-the-art solutions of pub/sub middleware in sensor-based networks, and compared their architectures, features and limitations, supported by comparative tables. As a reference model for any pub/sub middleware we propose a generic architecture that can be used as a reference to build new pub/sub solutions for sensor networks. Also, a comparative study for the most suitable simulators used for testing and evaluating WSN pub/sub solutions is presented in this work.

From the surveyed solutions, we can conclude that pub/sub solutions for limited resources networks still need more efforts in design, implementation, and testing phases. More concern still needed to consider the tradeoff of the middleware generality and the degree of application-specificity; this may lead to significant improvements in resources consumption. The proposed solutions also lack the efficient mechanisms to deal with the most impact factors on the performance of the pub/sub middleware such as churn rate, publish/subscribe rates, and failure rate. Moreover, ready testing and evaluating tools for pub/sub interaction models need to be taken in consideration in future research of modeling and simulation. For example, most of the proposed solutions did not evaluate the energy consumption, although, it is a very important measurement in evaluating sensor networks,

that is mostly due to the lack of the models and simulators in this area. This could be the reason behind the lack of energy efficient mechanisms in those solutions.

Applying pub/sub interaction model to simplify the development and integration of distributed systems will be at the expense of huge communication in the underlying layers. Therefore, it is not easy task to adapt such solution for limited resources systems such as WSN; to the best of our knowledge none of the proposed solutions have thoroughly investigated the actual cost in terms of memory, computation, communication, and energy consumption, we believe that this still needs considerable amount of effort to be dedicated.

A reliability protocol design and implementation for wireless sensor/actuator networks was introduced in this work. This protocol was integrated with TinyDDS middleware, and named as Reliable TinyDDS (RTDDS). The RTDDS design, implementation and performance evaluation were detailed in order to form an academic basic infrastructure for studying, testing, and improving reliability in WSN. RTDDS implementation prove that reliability QoS is applicable in most sensor platforms nowadays, since it is integrated and tested with middleware technology. As DDS is widely used nowadays, that makes RTDDS easily integrated to enterprise networks and increase the range of supported applications because of its flexibility in offering different reliability levels. The results show that RTDDS can work perfectly with applications that have time-sensitivity less than 5 seconds and half overloaded in terms of number of publishers. However, RTDDS still would be considered as a real-time system if it works with few nodes, where the response time would be in the range of few tens of milliseconds. Many research directions could improve RTDDS performance, or test its suitability in different network topologies and conditions. For example, RTDDS needs to be tested in random network topologies instead of grid

topology, mobile nodes, and secure environments. Moreover, instead of using fixed retransmission timeout, an adaptive retransmission timeout could be used and tested.

While EATDDS appears to be a promising middleware for WSN, there are still several enhancements that may make the middleware more widely applicable. In the current version of EATDDS, we assume sensors/publishers periodically transmit data to the base station/subscribers, i.e. one transmission per sensing data process. To save energy, sensors may work in the event-driven basis, where it only sends data when there is an event of interest, and this may lead to one important issue which is to port another well-known DDS-based QoS to the EATDDS, which is called Content-Based Filter (CBF). In this QoS, the node will filter the data by doing in-network check, if the reading is above or below certain threshold, then it is transmitted, otherwise keep monitoring. This QoS may significantly improve the efficiency of EATDDS, in terms of energy consumption. Also, if the data aggregation techniques are used, it may further minimize the total energy dissipation and end-to-end delay.

The main reason of using grid topology in our evaluation is to compare with the default TinyDDS, which uses 4 x 4 grid topology. Although the Grid topology are used in many indoor and outdoor applications, evaluating EATDDS using probabilistic topologies may raise new issues related to EATDDS performance and its implementation, for example energy consumption distribution is extremely dependent on the underlying routing protocols that is directly affected by the network topology. Therefore, one of the important future works is to evaluate the performance of EATDDS over probabilistic topologies, e.g. random, uniform, normal ... etc.

Furthermore, many parameters and timers are significantly affect he EATDDS performance such as round time, information gathering round, the synchronization between both of these missions, cluster formation approach, selecting the new RN node, for example could be not the maximum but above certain threshold. These parameters can be individually studied and improved.

References

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393-422, 2002.
- [2] M. Petrovic, V. Muthusamy and H.-A. Jacobsen, "Managing Automation Data Flows in Sensor/Actuator Networks," MSR/G, Toronto, November, 2007.
- [3] L. Atzori, A. Lera and G. Morabito, "The internet of things: A survey," *Computer Networks*, pp. 2787-2805, 2010.
- [4] G. Jorge, E. Monterio and J. Sa Silva, "Security in the integration of low-power Wireless Sensor Networks with the Internet: A survey," *Ad Hoc Networks*, pp. 264-287, 2015.
- [5] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad Hoc Networks*, vol. 2, no. 4, pp. 351-367, October 2004.
- [6] S. Oh, J.-H. Kim and F. Geoffrey, "Real-time performance analysis for publish/subscribe systems," *Future Generation Computer Systems*, vol. 26, no. 3, pp. 318-323, 2010.
- [7] P. T. Eugster, P. A. Felber, R. Guerraoui and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114-131, 2003.
- [8] E. Souto, G. Guimaraes, 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.
- [9] G. Cugola and H.-A. Jacobsen, "Using publish/subscribe middleware for mobile systems," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 4, pp. 25-33, October 2002.
- [10] A.-H. Jallad and T. Vladimirova, "Data-Centricity in Wireless Sensor Networks," *Computer Communications and Networks*, pp. 183-204, 2009.
- [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] S. Furrer, W. Schott, H.-L. Truong and B. Weiss, "The IBM wireless sensor networking testbed," in *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, TRIDENTCOM 2006*, 2006.
- [13] V. Tsetsos, G. Alyfantis, T. Hasiotis, O. Sekkas and S. Hadjiefthymiades, "Commercial wireless sensor networks: technical and business issues," in *IEEE Second Annual Conference on Wireless On-demand Network Systems and Services. WONS 2005*, 2005.
- [14] S. Schneider, "What Is Real-Time SOA?," RTI, Real-Time Innovations, Inc., 2010.
- [15] B. Krishnamachari, D. Estrin and S. Wicker, "Modelling data-centric routing in wireless sensor networks," in *IEEE infocom*, 2002.
- [16] D. A. Tran and L. H. Truong, "Enabling Publish/Subscribe Services in Sensor Networks," *Advances in Next Generation Services and Service Architectures*, 2011.
- [17] S. Taherian and B. Jean, "A publish/subscribe protocol for resource-awareness in wireless sensor networks," in *Proceeding of the international Workshop on Localized Algorithms and Protocols for Wireless Sensor Networks (LOCALGOS'07)*, 2007.
- [18] X. Tong 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.
- [19] 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.
- [20] 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. 4, pp. 1225-1239, July 2011.

- [21] 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.
- [22] B. University, "Tinydds: Publish/Subscribe Middleware for Wireless Sensor Networks," [Online]. Available: <https://code.google.com/p/tinydds/>. [Accessed 12 Feb. 2015].
- [23] Y. Liu and B. Plale, "Survey of publish subscribe event systems," 2003.
- [24] R. Baldoni, L. Querzoni and A. Virgillito, "Distributed event routing in publish/subscribe communication systems: a survey," 2005.
- [25] M. Berbineau, M. Jonsson, J.-M. Bonnin, S. Cherkaoui, M. Aguado, C. Rico-Garcia, H. Ghannoum, R. Mehmood and A. Vinel, "Survey on Context-Aware Publish/Subscribe Systems for VANET," in *Communication Technologies for Vehicles*, vol. 7865, Springer Berlin Heidelberg, 2013, pp. 46-58.
- [26] M.-M. Wang, J.-N. Cao, J. Li and S. K. Dasi, "Middleware for Wireless Sensor Networks: A Survey," *Journal of Computer Science and Technology*, vol. 23, no. 3, pp. 305-326, 2008.
- [27] S. Hadim and N. Mohamed, "Middleware for Wireless Sensor Networks: A Survey," in *First International Conference on Communication System Software and Middleware, IEEE Comsware 2006.*, 2006.
- [28] M. Molla and S. Ahamed, "A survey of middleware for sensor network and challenges," in *International Conference on Parallel Processing Workshops, ICPP 2006 Workshops, IEEE.*, 2006.
- [29] P. Boonma and J. Suzuki, "Middleware Support for Pluggable Non-Functional Properties in Wireless Sensor Networks," in *IEEE Congress on Services - Part I*, 2008.
- [30] P. Boonma and J. Suzuki, "Self-Configuring Publish/Subscribe Middleware for Wireless Sensor Networks," in *Consumer Communications and Networking Conference. CCNC 2009. 6th IEEE*, 2009.
- [31] J. Heidemann, F. Silva and D. Estrin, "Matching data dissemination algorithms to application requirements," in *SenSys '03 Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003.

- [32] C. Esposito, D. Cotroneo and S. Russo, "On reliability in publish/subscribe services," *Computer Networks*, vol. 47, 2013.
- [33] Y. Huang and H. Garcia-Molina, "Publish/Subscribe in a Mobile Environment," *Wireless Networks*, pp. 643-652, 2004.
- [34] L. Luo, T. He, G. Zhou, L. Gu, T. Abdelzaher and J. Stankovic, "Achieving Repeatability of Asynchronous Events in Wireless Sensor Networks with EnviroLog," in *25th IEEE International Conference on Computer Communications. Proceedings INFOCOM 2006*, 2006.
- [35] M.-M. Wang, J.-N. Cao, J. Li and S. K. Das, "Middleware for wireless sensor networks: A survey.," *Journal of computer science and technology*, vol. 23, no. 3, pp. 305-326, 2008.
- [36] P. Levis and D. Culler, "Mate : a tiny virtual machine for sensor networks," in *ASPLOS X Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, 2002.
- [37] P. Levis, D. Gay and D. Culler, "Bridging the Gap: Programming sensor networks with application specific virtual machines," in *Proc. the 6th Symp. Operating Systems Design and Implementation (OSDI 04)*, San Francisco, USA, 2004.
- [38] R. Gummadi, O. Gnawali and R. Govindan, "Macro-programming wireless sensor networks using kairos," *Distributed Computing in Sensor Systems*, pp. 126-140, 2005.
- [39] M. Welsh and G. Mainland, "Programming sensor networks using abstract regions," in *Proc. the 1st Usenix/ACM Symp. Networked Systems Design and Implementation (NSDI 04)*, San Francisco, CA, March, 2004.
- [40] R. Lewis, *Advanced Messageing Applications with MSMQ and MQSeries*, Que, 1999.
- [41] OMG, "Data Distribution Services (DDS)," 1 1 2007. [Online]. Available: <http://www.omg.org/spec/DDS/>. [Accessed October 2013].
- [42] A. S. Stanford-Clark and H. L. Truong, "MQTT for sensor networks (MQTT-S) protocol specification," 2008.
- [43] TIBCO, "TIBCO Rendezvous," Software Release 8.3.0, 2010.

- [44] M. Sun, "Java™ Message Service," 9 November 1999. [Online]. Available: <http://docs.oracle.com/cd/E19957-01/816-5904-10/816-5904-10.pdf>. [Accessed October 2013].
- [45] OMG, "CORBA notification service specification," 11 OCT. 2004. [Online]. Available: <http://www.omg.org/spec/NOT/1.1/>. [Accessed October 2013].
- [46] OMG, "CORBA event service specification," 2004. [Online]. Available: <http://www.omg.org/spec/EVNT/1.2/>. [Accessed October 2013].
- [47] R. Bastide, O. Sy, D. Navarre and P. Palanque, "A formal specification of the CORBA event service," in *Fourth International Conference on Formal methods for open object-based distributed systems*, 2000.
- [48] M. Castro, P. Druschel, A. Kermarrec and A. Rowston, "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1489-1499, October 2002.
- [49] B. Oki, M. Pfluegl, A. Siegel and D. Skeen, "The Information Bus: an architecture for extensible distributed systems," vol. 27, no. 5, pp. 58-68, Dec 1993.
- [50] S. Baehni, P. T. Eugster and R. Guerraoui, "Data-aware multicast," in *International Conference on Dependable Systems and Networks*, 2004.
- [51] G. Pardo-Castellote, "OMG Data-Distribution Service: Architectural Overview," in *Proceedings of the 23 rd International Conference on Distributed Computing Systems Workshops*, 2003.
- [52] RTI, "RTI Connex DDS," 2013. [Online]. Available: <http://www.rti.com/products/dds/index.html>. [Accessed October 2013].
- [53] M. Altherr, M. Erzberg and S. Maffeis, "iBus - a software bus middleware for the java platform," in *Proceedings of the International Workshop on Reliable Middleware Systems*, 1999.
- [54] S. Deering, "Host Extentsions for IP Multicasting," 1989.
- [55] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, no. 2, pp. 85 - 110, May 1990.

- [56] S. Banerjee, B. Bhattacharjee and C. Kommareddy, "Scalable application layer multicast," in *SIGCOMM '02 Proceedings of the 2002 ACM conference on Applications, technologies, architectures, and protocols for computer communications*, 2002.
- [57] J. Jannotti, D. K. Gifford, K. L. Johnson and M. F. Kaashoek, "In Proceedings of the 4th conference on Symposium on Operating System Design & Implementation," in *Overcast: reliable multicasting with an overlay network*, 2000.
- [58] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Sturman and D. Sturman, "Exploiting IP multicast in content-based publish-subscribe systems," in *IFIP/ACM International Conference on Distributed systems platforms*, New York, 2000.
- [59] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. Strom and D. Sturman, "An efficient multicast protocol for content-based publish-subscribe systems," in *In Proc. of the 19th International Conference on Distributed Computing Systems*, 1999.
- [60] A. Cheung and H.-A. Jacobsen, "Load Balancing Content-Based Publish/Subscribe Systems," *ACM Transactions on Computer Systems*, vol. 28, no. 4, December 2010.
- [61] A. Carzaniga, D. Rosenblum and A. Wolf, "Design and Evaluation of a Wide-Area Notification Service," *ACM Transactions on Computer Systems (TOCS)*, vol. 19, no. 3, p. 332–383, Aug 2001.
- [62] F. Fabret, A. Jacobsen, F. Llirbat, J. Pereira, K. Ross and D. Shasha, "Filtering algorithms and implementation for very fast publish/subscribe systems," *ACM SIGMOD Record*, vol. 30, no. 2, p. 115–126, 2001.
- [63] A. Carzaniga and A. Wolf, "Forwarding in a content-based network," in *SIGCOMM '03 Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003.
- [64] G. Li, Y. Wang and J. Feng, "Location-aware publish/subscribe," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, 2013.
- [65] A. Margara and G. Cugola, "High Performance Publish-Subscribe Matching Using Parallel Hardware," *IEEE Transactions on Parallel and Distributed Systems*, 2013.

- [66] D. Tran and C. Pham, "DIBS: Efficient distributed information brokerage in large-scale sensor networks," *Ad Hoc Networks*, vol. 11, no. 3, p. 735–746, 2013.
- [67] A. González, W. Mata, L. Villaseñor, R. Aquino, J. Simo, M. Chávez and A. Crespo, "μDDS: A Middleware for Real-time Wireless Embedded Systems," *Journal of Intelligent & Robotic Systems*, vol. 64, no. 3-4, pp. 489-503, December 2011.
- [68] C. P. Hall, A. Carzaniga, J. Rose and A. L. Wolf, "A Content-Based Networking Protocol for Sensor Networks," 2004.
- [69] K. Shi, Z. Deng and X. Qin, "TinyMQ: A content-based publish/subscribe middleware for wireless sensor networks," in *SENSORCOMM 2011, The Fifth International Conference on Sensor Technologies and Applications*, 2011.
- [70] P. Eugester, "Type-based publish/subscribe: Concepts and experiences," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 29, no. 1, pp. 1-50, January 2007.
- [71] P. Eugster, "TYPE-BASED PUBLISH/SUBSCRIBE," 2001.
- [72] B. Wang, P. Liu, G. Wang and X. Zhao, "ELM based approximate dynamic cycle matching for homogeneous symmetric Pub/Sub system," *World Wide Web*, June 2013.
- [73] R. Baldoni, C. Marchetti, A. Virgillito and R. Vitenberg, "Content-Based Publish-Subscribe over Structured Overlay Networks," in *Proceedings. 25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005)*, 2005.
- [74] X. Ma, Y. Wang, Q. Qiu, W. Sun and X. Pei, "Scalable and elastic event matching for attribute-based publish/subscribe systems," *Future Generation Computer Systems*, September 2013.
- [75] W. Rjaibi, K. Dittrich and D. Jaepel, "Event matching in symmetric subscription systems," in *CASCON '02 Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research*, 2002.
- [76] T. W. Yan and H. García-Molina, "Index structures for selective dissemination of information under the boolean model," *ACM Transactions on Database Systems (TODS)*, vol. 19, no. 2, pp. 332-364, 1994.
- [77] J. Pereira, F. Fabret, F. Llirbat and D. Shasha, "Efficient matching for web-based publish/subscribe systems," *Cooperative Information Systems*, pp. 162-173, 2000.

- [78] M. Aguilera, R. Strom, D. Sturman, M. Astley and T. Chandra, "Matching Events in a Content-Based Subscription System," in *Proceedings of The ACM Symposium on Principles of Distributed Computing (PODC 1999)*, 1999.
- [79] A. Campailla, S. Chaki, E. Clarke, S. Jha and H. Veith, "Efficient filtering in publish-subscribe systems using binary decision diagrams Software Engineering," in *Proceedings of the 23rd International Conference on*, 2001.
- [80] J.-H. Hauer, V. Handziski, A. Kopke, 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. 49, no. 13, pp. 369-385, 2008.
- [81] P. Costa, G. Picco and S. Rossetto, "Publish-subscribe on sensor networks: a semi-probabilistic approach," in *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, 2005.
- [82] H. Alnuweiri, M. Rebai and R. Beraldi, "Network-coding based event diffusion for wireless networks using semi-broadcasting," *Ad Hoc Networks*, vol. 10, no. 6, p. 871-885, 2012.
- [83] A. Boukerche, X. Cheng and J. Linus, "A Performance Evaluation of a Novel Energy-Aware Data-Centric Routing Algorithm in Wireless Sensor," *Wireless Networks*, vol. 11, no. 5, pp. 619-635, 2005.
- [84] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325-349, May 2005.
- [85] A. Carzaniga, M. Rutherford and A. Wolf, "A routing scheme for content-based networking," in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004.
- [86] J. Martins and S. Duarte, "Routing algorithms for content-based publish/subscribe systems," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 1, pp. 39-58, First Quarter 2010.
- [87] A. Corsaro, L. Querzoni, S. Scipioni, S. Piergiovanni and A. Virgillito, "Quality of service in publish/subscribe middleware," in *Emerging Communication: Studies in New Technologies and Practices in Communication*, vol. 8, Global Data Management, 2006, pp. 79-97.

- [88] D. Chen and P. K. Varshney, "QoS Support in Wireless Sensor Networks: A Survey," in *International Conference on Wireless Networks*, 2004.
- [89] M. Sharifi, M. Taleghan and A. Taherkordi, "A Middleware Layer Mechanism for QoS Support in Wireless Sensor Networks," in *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies. ICN/ICONS/MCL 2006.*, 2006.
- [90] F. Xia, "QoS Challenges and Opportunities in Wireless Sensor/Actuator Networks," *Sensors*, vol. 8, no. 2, pp. 1099-1110, 2008.
- [91] T. Rault, A. Bouabdallah and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, no. 4, p. 104–122, July 2014.
- [92] M. Anisi, A.-H. Abdullah and S. Razak, "Energy-efficient and reliable data delivery in wireless sensor networks," *Wireless Networks*, vol. 19, no. 4, pp. 495-505, 2013, Volume 19, Issue 4, pp 495-505 May 2013.
- [93] 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.
- [94] 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.
- [95] 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.
- [96] A. Stanford-Clark and U. Hunkeler, "MQ Telemetry Transport (MQTT)," 1999. [Online]. Available: <http://mqtt.org>. [Accessed 22 9 2013].
- [97] 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. 2013].

- [98] 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.
- [99] A. S. Tanenbaum, *Computer Networks*, Boston: Prentice Hall PTR, 2011.
- [100] D. A. Tran and C. Pham, "A content-guided publish/subscribe mechanism for sensor networks without location information," *Computer Communications*, vol. 33, no. 13, pp. 1515-1523, 16 August 2010.
- [101] D. A. Tran and C. Pham, "PUB-2-SUB: A Content-Based Publish/Subscribe Framework for Cooperative P2P Networks," in *NETWORKING '09 Proceedings of the 8th International IFIP-TC 6 Networking Conference*, 2009.
- [102] P. Eugster and R. Guerraoui, "Probabilistic multicast," in *Proceedings. International Conference on Dependable Systems and Networks, DSN 2002*, 2002.
- [103] H. Sundani, H. Li, V. Devabhaktuni, M. Alam and P. Bhattacharya, "Wireless Sensor Network Simulators A Survey and Comparisons," *International Journal of Computer Networks (IJCN)*, vol. 2, no. 5, pp. 249-265, 2011.
- [104] X. Xian, W. Shi and H. Huang, "Comparison of OMNET++ and other simulator for WSN simulation," in *3rd IEEE Conference on Industrial Electronics and Applications. ICIEA 2008.*, 2008.
- [105] M. Jevtić, N. Zogović and G. Dimić , "Evaluation of Wireless Sensor Network Simulators," in *17th Telecommunications forum TELFOR 2009*, Serbia, Belgrade, 2009.
- [106] A. Dunkels, B. Gronvall and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks, 2004*, 2004.
- [107] V. Handziski, A. Kopke, A. Willig and A. Wolisz, "TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in *REALMAN '06 Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, 2006.
- [108] TOSSIM, "TinyOS Documentation Wiki," 2003. [Online]. Available: <http://docs.tinyos.net/index.php/TOSSIM>. [Accessed 24 Nov. 2013].

- [109] 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.
- [110] 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 3rd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, 2008.
- [111] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Proceedings 2006 31st IEEE Conference on Local Computer Networks*, 2006.
- [112] X. Chang, "Network simulations with OPNET," in *Simulation Conference Proceedings*, 1999.
- [113] NS2, "The Network Simulator- ns2," [Online]. Available: <http://www.isi.edu/nsnam/ns/>. [Accessed 25 Nov. 2013].
- [114] X. Zeng, R. Bagrodia and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," in *Proceedings. Twelfth Workshop on Parallel and Distributed Simulation, 1998. PADS 98*, 1998.
- [115] Castalia, "Wireless Sensor Network Simulator, Castalia," [Online]. Available: <http://castalia.research.nicta.com.au/index.php/en/>. [Accessed 25 Nov. 2013].
- [116] A. Boulis, "Castalia." A simulator for wireless sensor networks and body area networks," 2011.
- [117] I. Chakeres and C. Perkins, "Dynamic MANET on demand (DYMO) routing," in *Internet-Draft Version 17, IETF*, 2006.
- [118] 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.
- [119] 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 3rd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, 2008.

- [120] G. Benelli and A. Garzelli, "New modified stop-and-wait ARQ protocols for mobile communications," *Wireless Personal Communications*, vol. 1, no. 2, pp. 117-126, 1994.
- [121] A. Stanford-Clark and H. L. Truong, "MQTT For Sensor Networks (MQTT-SN)," IBM, 2013.
- [122] memsic, "TELOSB," memsic, [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf. [Accessed 27 1 2015].
- [123] T. INSTRUMENTS, "CHIPCON CC2420," TEXAS INSTRUMENTS, [Online]. Available: <http://www.ti.com/product/cc2420>. [Accessed 31 1 2015].
- [124] J. Polastre, R. Szewczyk and D. Culler, "Telos: enabling ultra-low power wireless research," in *Fourth International Symposium on Information Processing in Sensor Networks*, 2005.
- [125] 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.
- [126] M. Memsic, "MICAz," [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf. [Accessed 23 4 2015].
- [127] D. Gay, P. Levis, D. Culler and E. Brewer, "nesC 1.2 Language Reference Manual," TinyOS, 2005.
- [128] 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.
- [129] "TELOSB Crossbow," Crossbow, [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf. [Accessed 9 11 2014].

APPENDIX A

ID-Based Routing

This protocol is what we used as an underlying routing protocol in most of our simulations and empirical studies.

Flooding routing protocols incur significant amount of routing overhead, which leads to more resources consumption especially for constrained devices, e.g. sensors. Besides, there are many applications that are based on deterministic deployment for network nodes, such as indoor applications. For example, home, building, and factory automation and monitoring applications. Therefore, in such applications the flooding routing overhead can be avoided by using location-based routing. However, location-based routing also needs hardware support, i.e. GPS devices, which increase the sensor price. In this work, we propose an Id-based routing for Wireless Sensor Networks (WSN) based on nodes identifications; It is specialized for grid topology based applications. This protocol consumes almost zero memory footprint, where it does not need any memory space to save routing tables or even individual routes. Furthermore, it does not also need to send routing requests and replies for establishing routes, which consumes much energy due to radio sending/receiving of the routing packets.

DESCRIPTION

ID-based routing protocol is intended for WSN $M \times N$ grid topology. Unlike tradition WSN routing protocols, where the data is routed from the Base station to the sensor or vice versa, ID-based is free addressing routing protocol. That means, the source and destination can be any node in the network, as an example for such routing protocol is TYMO for WSN.

In ID-based protocol, there are several assumptions to work properly, these assumptions are as follows:

1. The topology is grid with any size of $M \times N$
2. The nodes have known identifications
3. The nodes are ordered, see figure 1.
4. All the node neighbors are in its transmission range, including the nodes in the diagonal directions, e.g. node 5 neighbors are 0,1,2,4,6,8,9, and 10, as shown in Figure 68.

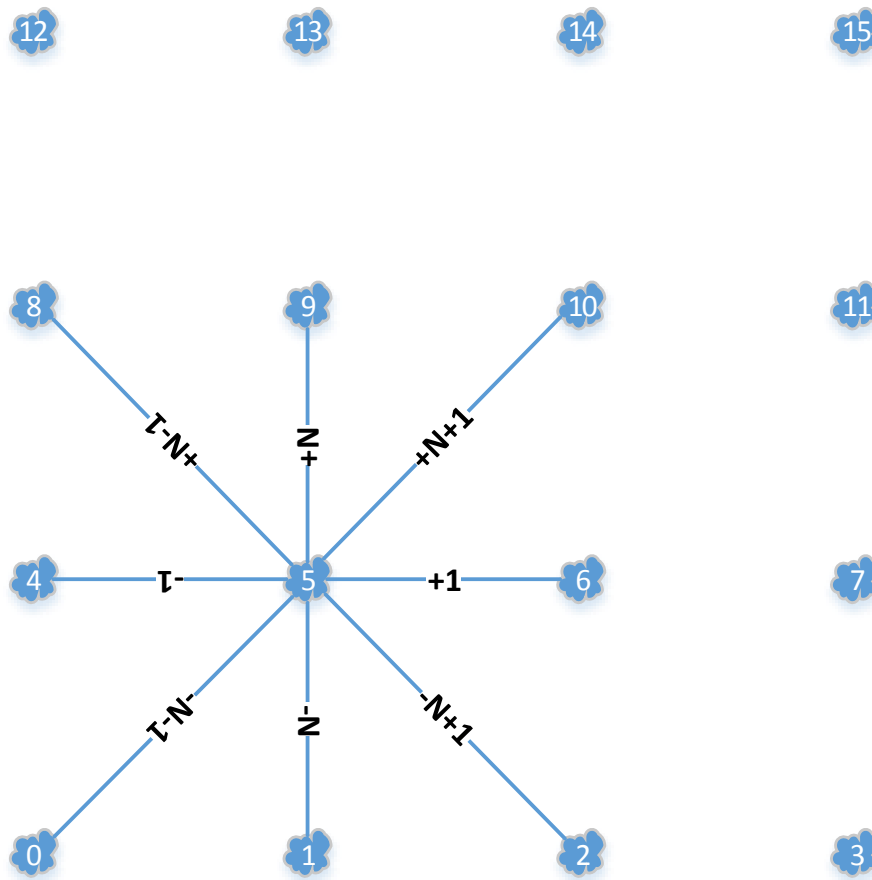


Figure 68: Id-based Routing method

As shown in Figure 68, any node in the network can exactly know in which direction it should go to reach its final destination. For example, suppose that the node number 5 is the source, and it needs to go to node 15, the final destination. Then if it goes diagonally it can reach the destination with two hops through node 10, which is the shortest path.

The protocol algorithm works as follows:

1. The sender examines the destination address to know its location, i.e. its row and column within the grid; we refer to them as D_R and D_C respectively.
2. The D_C is calculated by using modulus function as follows:

$$D_C = \text{dest} \bmod N; \text{ where dest is the destination id,}$$

3. The D_R is calculated by using floor function as follows:

$$D_R = \text{floor}(\text{dest}/N)$$

4. Using the obtained destination location (D_R, D_C), the sender or forwarder can easily compute the next-hop as follows:

- a. If ($D_C = S_C$) and ($Did > Sid$) then next-hop = $Sid + N$
- b. If ($D_C = S_C$) and ($Did < Sid$) then next-hop = $Sid - N$
- c. If ($D_R = S_R$) and ($Did > Sid$) then next-hop = $Sid + 1$
- d. If ($D_R = S_R$) and ($Did < Sid$) then next-hop = $Sid - 1$

If none of the above is true then:

- e. If ($D_C > S_C$) and ($Did > Sid$) then next-hop = $Sid + N + 1$
- f. If ($D_C > S_C$) and ($Did < Sid$) then next-hop = $Sid - N + 1$
- g. If ($D_C < S_C$) and ($Did > Sid$) then next-hop = $Sid + N - 1$
- h. If ($D_C < S_C$) and ($Did < Sid$) then next-hop = $Sid - N - 1$

5. Step number 4 will be repeated until next-hop = dest

Notice that the sender and forwarder use the same algorithm to forward the packet to the next hop. Furthermore, to ensure the shortest path, the route moves diagonally until it reaches the destination row or column then it moves horizontally or vertically respectively.

The second module of Id-Routing protocol is the maintenance module. We make as simple as the first module, where it also uses almost zero overhead. This module exploits the promiscuous mode of the WSN, where in this mode the node can silently listen to the transmission of its neighbor. Thereby, the node can listen to the transmission of its neighbor, in case it is not a final destination, and then it can ensure that the packet has been forwarded or not. In case of not forwarded the node resend the packet for maximum retrials and then it changes the path by selecting the next shortest path.

APPENDIX B

Adaptive Reliability Protocol for Wireless Sensor Networks using Packet

Delivery Ratio Metric

This is another proposed reliability protocol for WSN that may put the protocol overhead on the subscriber node rather than publisher node. This is effective for sensor-based networks since in practice the subscriber (base station) is often more powerful than the ordinary sensor node.

Providing reliability Quality of Service (QoS) to Wireless Sensor Networks (WSN) has a significant impact on the network performance and lifetime. That is because of two main factors: 1) the limited resources in sensor networks, such as memory, CPU, bandwidth, and energy. 2) the extreme overhead of operating a reliable QoS. Therefore, implementing reliability on sensor networks needs an efficient design and implementation. Unlike the strict reliable applications, such as military and healthcare applications, some of the sensor applications required a minimum level of reliability to achieve a specific degree of accuracy. Therefore, an adaptive reliability QoS is a potential solution in this case, where an efficient switching between the best-effort and reliable services may lead to significant savings in the WSN resources. In this work, we present an adaptive reliability protocol that suites sensor networks requirements and provides an efficient adaptive reliability support to the WSN applications.

Introduction

The Adaptive Reliability Quality of Service (ARQoS) protocol is designed to operate at the middleware or application layer, independently from the underlying layers' protocols. This work aims to provide an ARQoS to the WSN. It gives the receiver the ability to do an agreement with the sender to support a certain level of reliable QoS. For example, the receiver can agree with the sender to do not go under 90% of Packet Delivery Ration (PDR). In this work, we use the PDR performance metric as a reference to switch between reliability modes. The PDR is defined as the total successfully received packets divided by the total sent packets. The receiver side, e.g. base station, is responsible for calculating the PDR at a predetermined time interval (T). Each received packet is distinguished by its originator address and Packet Serial Number (PSN). The receiver uses these packet information to count the dropped packets during T , and hence calculates the PDR of T period. If the resulted PDR is less than the requested PDR percentage, then a switch message is sent to the sender to switch to reliable mode. As soon as PDR returns to the desired value, a switch message is send again to the sender to switch from reliable mode to best effort mode, and so on. One bit in the message header is used to indicate the reliability mode, which is either reliable or best effort.

The ARQoS policy has three supported levels, ranging from low to high reliability, viz., best effort, adaptive, and reliable. By using a special type of packet, the receiver can request a specific reliability level from the sender by sending the reliability percentage value at session initiation, or while it is running. This value is used to distinguish between the supported three levels. The first level is the best effort level that usually suits the time-sensitive applications and is represented by the zero reliability value (0%). In this level, no acknowledgments are used and hence neither calculations are needed at the receiver side

nor buffering at the sender side. The second is the adaptive level that is represented by the reliability percentage value which is less than 100% and more than zero%. This level is the essence of this work and it uses the PDR metric to adapt a reliable QoS. The last level is the reliable level, which suits the data-sensitive level and is represented by the reliability percentage value of 100%. In this level, we use the NACK method to minimize the reliability overhead. The requested level of reliability is application-specific, and most likely depends on the tracked or monitored object's changing rate. For example, monitoring the weather is most likely to use the best effort reliability level because the weather changing rate is very slow, and the sampling rate is usually in minutes or even hours. In contrast, in military applications, tracking a rocket by defense systems needs the highest reliability level, i.e. 100%, and sampling rate in the range of millisecond or even in microseconds.

Algorithm Description

Since the base station typically has an infinite energy source, it is more appropriate to build the loss and switching control in there. In this algorithm, we describe how adaptive switching may be implemented to realize adaptive reliability. The only modification in the packet header that is needed to implement this protocol is to add a mode bit. This bit is used in the switching mechanism to switch between the system modes. At the receiver side, during the operation time, there are periodic tests that monitor the system reliability by measuring the packet delivery ratio (PDR). These periodic tests are referred to in this document by rounds. In Figure 69, we show the round period and the time interval (TI) that separates the rounds. Determining these times efficiently has a significant impact on the overall application performance.

For simplicity, the algorithm is divided into two sub-algorithms, viz. the round algorithm and the switching algorithm. In Figure 70, we describe the round algorithm steps, and show how it counts the number of successfully received packets and dropped packets by using the PSN, and Last PSN (LPSN) values. In each round, the received and dropped packets are counted and then submitted to the switching algorithm to calculate the PDR and make a decision to whether to change the mode. The switching algorithm steps is depicted in Figure 71. In this figure, the round processing step refers to the algorithm specified in Figure 70. In the switching algorithm, the PDR is calculated and checked against the requested reliability level. If the current PDR is less than the requested reliability level and the mode was in best-effort, then the protocol switches to reliable mode. Vice versa, if it is larger than the requested reliability level, and the mode was in the reliable level, then it switches to best-effort level, and so on.

Two main mechanisms are added to both of these algorithms to increase their efficiency. The first one is the Assurance Time Interval (ATI), which is a period of time that is placed just at the end of every round period, as shown in Figure 69. The purpose of the ATI is to assure the reception of all sent packets that are relevant to the round packets, i.e. all the packets with PSN less than the LPSN received at the end of the round. The second mechanism is doubling the threshold mechanism that is used to minimize the switching overhead as depicted in Figure 72. Double threshold is used to mitigate the switching overhead due to the network instability. These thresholds are called upper and lower thresholds, where the upper threshold is used to switch from reliable to best-effort mode; and vice versa. The upper/lower threshold values can be determined statically during the initial stage of the deployment phase. Alternatively; these values may be determined

dynamically during the application operation time; where in this case they vary based on the dynamic network conditions. During the network life-time, the application is running at one of the switching regions: the best-effort region, the switch region, or the reliable region. The three regions are depicted in Figure 72 where the switching region is the region that is bounded by the upper and lower thresholds.

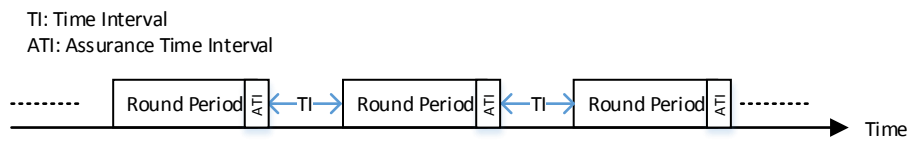


Figure 69 Rounds distribution over the network life time

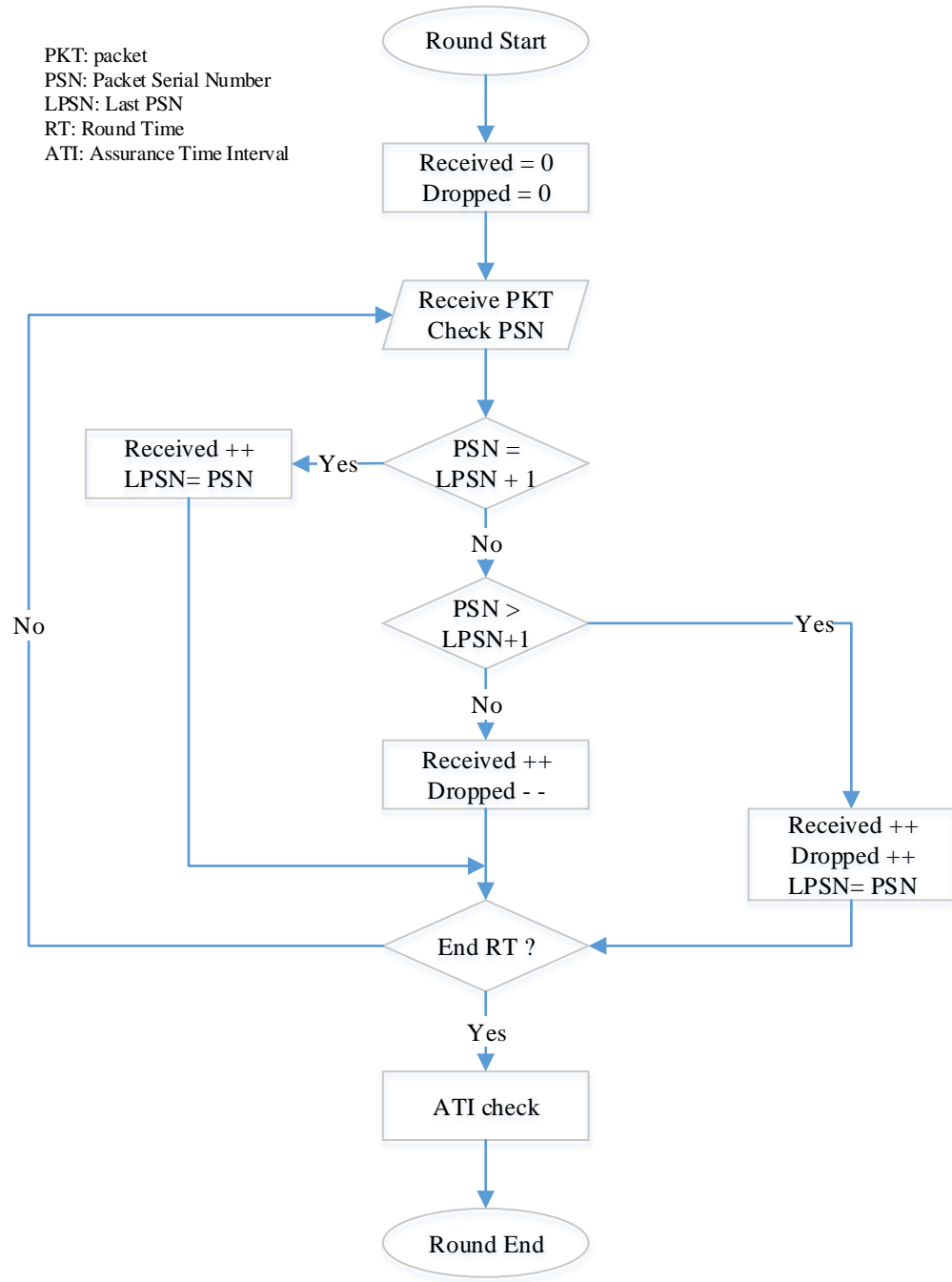


Figure 70 Round algorithm flowchart

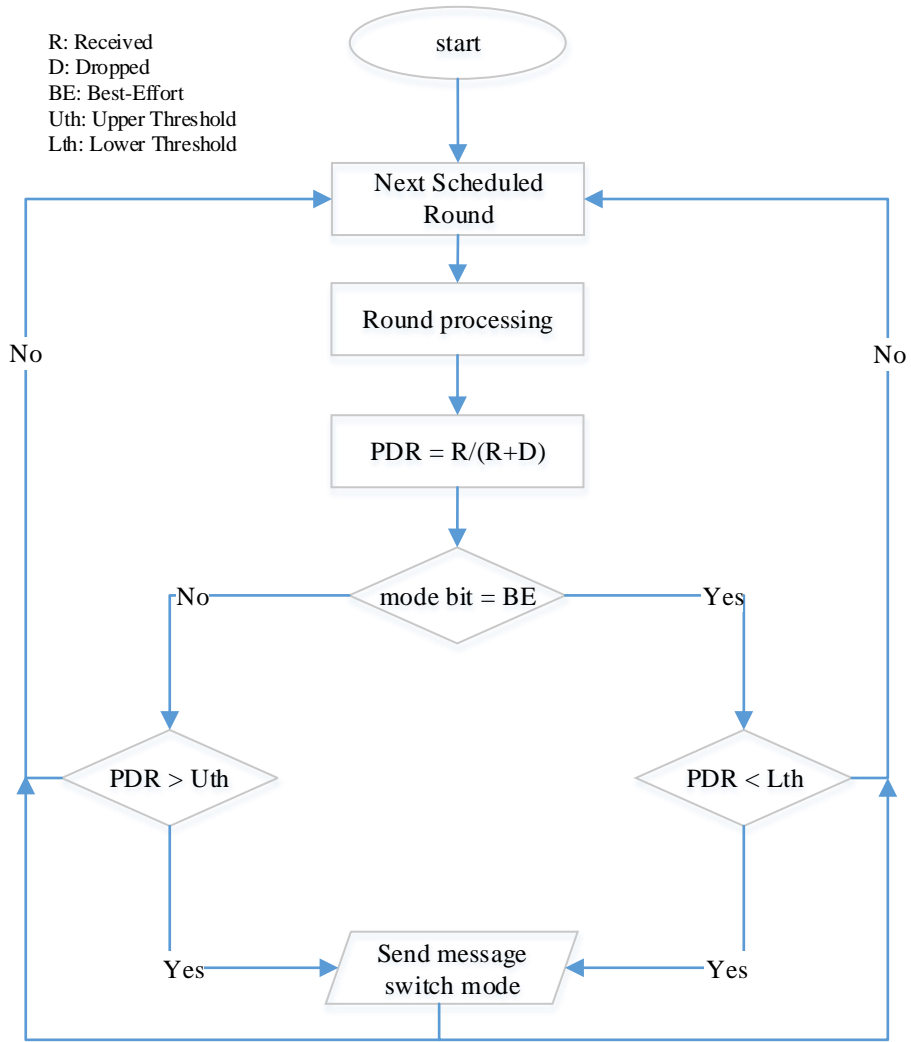


Figure 71 Adaptive reliability switching algorithm flowchart

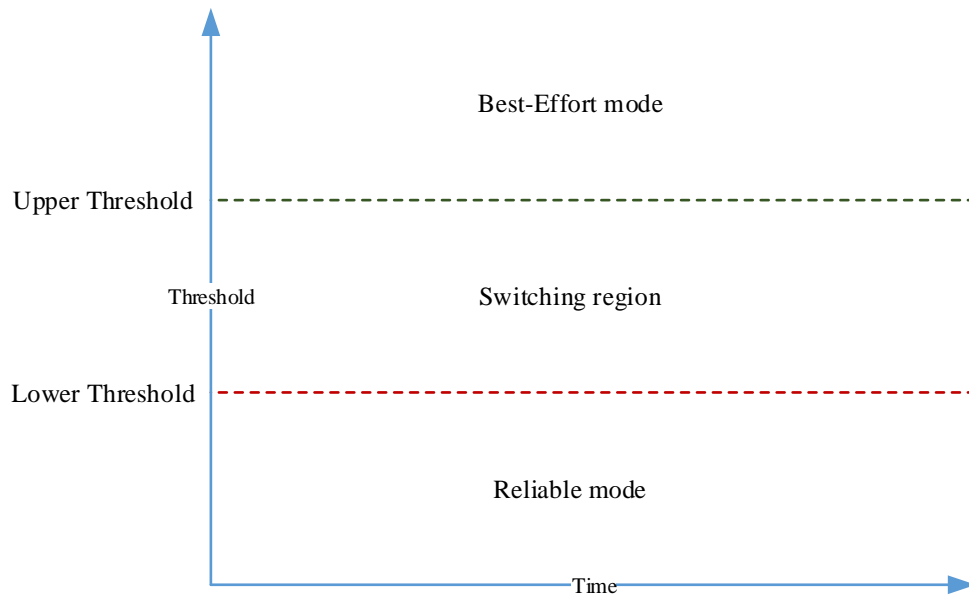


Figure 72 Adaptive reliability regions

Vitae

Anas A. Hasan Al-Roubaiey - PhD in Computer Networks

Personal

Date of Birth: 22/May/1976

Nationality: Yemeni

Mobil:+9(665) 0562-4163

P.O.Box 1498, Dhahran, 31261,

Saudi Arabia

Academic Email:

roubaiey@kfupm.edu.sa

Personal Email:

saba717671@hotmail.com

University

PhD In Computer Networks

Department of Computer Engineering

King Fahd University of Petroleum &
Minerals

Phone: +966 3 860-1423

Fax: +966 3 860-3059

Web:

[https://www.researchgate.net/profile/
Anas_Al-
Roubaiey/contributions?ev=prf_act](https://www.researchgate.net/profile/Anas_Al-Roubaiey/contributions?ev=prf_act)

Education

Defended

On May 3, 2015. **PHD IN COMPUTER SCIENCE AND ENGINEERING.**

**“ENERGY AWARE MIDDLEWARE FOR WIRELESS
SENSOR/ACTUATOR NETWORKS”**

**King Fahd University of Petroleum and Minerals (KFUPM),
Dhahran, Saudi Arabia**

**Supervisor: Dr. Tarek Sheltami, leading researcher in the area of
ad hoc and sensor networks.**

The target of the study is to develop an energy aware middleware for wireless sensor and actuator networks. Our work was based on the DDS (Data Distribution Service) standard, we called our proposed technique EATDDS (Energy Aware TinyDDS).

Experimental tests are being conducted to evaluate the exact overhead of adding middleware to limited resources devices such as sensors/actuators. TinyOS and TOSSIM simulator are used in this study, where I build the scenarios and test them on the TOSSIM simulator; then I download the code to real TelosB sensors to take empirical results. In this work, several languages are being used such as nesC, java, python, awk, MATLAB, and C++.

2005-2009

**M.S IN COMPUTER NETWORKS. “INTRUSION
DETECTION IN MOBILE AD-HOC NETWORKS:**

IMPLEMENTATION AND PERFORMANCE EVALUATION OF ADAPTIVE ACKNOWLEDGMENT APPROACH”

**King Fahd University of Petroleum and Minerals (KFUPM),
Dhahran, Saudi Arabia**

Supervisor: Dr. Tarek Sheltami.

The target of the study was to develop a new intrusion detection technique for mobile ad hoc networks based on TwoACK technique. We enhanced this technique by adding end-to-end acknowledgment and improving its detection precision. We called it AACK (Adaptive ACKnowledgment). Recently, it has been improved by EAACK technique that is published in IEEE Transactions on Industrial Electronics journal. Based on google scholar our key paper yet gained 72 citation.

Extensive simulation tests were conducted to evaluate the performance of AACK intrusion detection system. I used network simulator NS2 to implement the AACK technique over DSR routing protocol. The scenarios were a combination of different node speeds, data rates, and network density.

1996-2000

B.S IN COMPUTER ENGINEERING.

**Arab Academy for Science & Technology University, Alexandria,
Egypt**

I got my B.S degree based on grant from Ministry of Higher education in Yemen as a result of my performance in High school.

My graduation project involved design of an automated farm system. Where I built a prototype that included some sensors, e.g. light, temperature,

humidity. The farm is automatically managed, where, as an example, it has an automated cover that is closed when the temperature exceeded certain degree. I used a C language for data acquisition and control.

The project grade was Distinction.

In my B.S Degree, I got the third honor out of 70 student (total number of students in the College of Engineering)

Teaching and administrative experience

2005-2014

King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia

Since I joined KFUPM in 2005, I have been involved in teaching different kind of courses for both undergraduate and graduate levels.

In the teaching activates, the courses that I have helped in teaching include:

1. Undergraduate level:
 - Introduction to C++
 - Embedded systems LAB (ARM)
2. Graduate level:
 - NS2 Simulator for Ad-Hoc networks course
 - Heterogeneous Computing LAB

In some of those courses (undergraduate and graduate level), I had to do a grading work, help with projects and supervision, exam grading, and one-to-one student support. In the graduate level course, the work was more complicated and involved, in which I had to deeply support the students in their advanced research projects.

Lately, I have been assigned to be the Vice-President of the Graduate Student Club (GSC) at KFUPM. The work included preparation of many activities including academic and social activities.

2001-2004

Taiz University, Taiz, Yemen

I was assigned as a lecturer assistant in Taiz University, Yemen in 2001. I was involved in teaching different kind of courses for undergrad students including:

- Introduction to Computer.
- C++ language.
- Object Oriented Programming.
- Computer Programs for Engineers.

In some courses, I had to do lectures, labs, tutorials, and one-to-one student support in the student's coursework projects.

2002-2004

IT department of Ministry of Interior, Sana'a, Yemen

I was involved in a big project as a team member (Vice-president of the team) in design and implementation of Yemeni ports network. In this project we built a full network that communicates the all ports in Yemen to the IT center in Ministry of Interior.

During this project, I went to USA for training for two weeks. In this training I took short course in Borders application and how to install WAN networks.

RESEARCH

From my education section, my research interests are mainly in wireless sensor and ad hoc networks, distributed systems, middleware, intrusion detection systems, multimedia.

During my study in KFUPM I have involved in three funded project as follows:

- KFUPM No. IN070377, Maximizing the Number of Hops in Video Streaming over Mobile Ad Hoc Networks using Artificial Intelligence, 2009. Member.

-
- NSTIP No. 09ELE04785, Wireless Stress Indicator sensor for condition based Monitoring in e-maintenance, Member, 2011 – 2013.
 - NSTIP. No. 12-ELE2381-04, Efficient Implementation of Non-Intrusive Leak Detection System, member, 2013 – 2015.

Mainly I was working on simulations, prototyping and writing some of the published papers. Recently, I have also prepared two NSTIP project proposals and have been submitted to KFUPM NSTIP projects office. Those proposals was about using DDS middleware in sensor networks and using DDS middleware in Oil and Gas industry.

Now I am working also on KFUPM Internal fund project number RG1319-1, my role is test and improve a time synchronization protocols in sensor networks using TOSSIM simulator and TelosB platform.

Honors

- 3rd position of Bachelor degree in Computer Engineering
- 1st place in the university chess championship, Egypt, 2000
- Cisco Certifications CCSA1 and CCSA2 with honor letters
- Outstanding reviewer, in the top 10%, from AD HOC NETWORKS journal IF= 1.9, 2014

CONFERENCES AND WORKSHOPS

- IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2012.
 - 2nd International Conference on Manufacture Engineering, Quality and Production System (ICMEQP 2013), Hong Kong, China, 2013.
 - The Sixth International Symposium on Applications of Ad hoc and Sensor Networks (AASNET'14) in conjunction with the 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2014) in Halifax, Nova Scotia, Canada on September 22-25, 2014.
 - The IEEE International Wireless Communications and Mobile Computing Conference (IWCMC 2014) on August 4-8, Cyprus, 2014.
-

I attended those conferences in China as an author. I have presented two of my researches that are listed in the publications list. The last symposium (AASNET'14), I am participating as a publicity chair committee member, and also as a presenter for my research.

TECHNICAL AND PROGRAMS KNOWLEDGE

- Linux OS.
 - Java, C, C++, C#
 - NS2 Simulator
 - TOSSIM Simulator
 - TinyOS and Contiki Sensors OS
 - TelosB, Micaz, Arduino Sensor platforms
 - MATLAB 2012
 - LabVIEW
 - DDS Middleware
 - RTI DDS Tools
 - AWK (scripting language)
 - Python Language
 - NesC Component based programming language
-

LANGUAGES

- Arabic – native language
 - English – speak fluently and read/write
-

MEMBERSHIPS

- 2012 - now
- IEEE student member
-

PUBLICATIONS AND PATENTS

Patents

1. **Anas Al-Roubaiey**, Uthman Baroudi, “Method for Determining Leak Location in Pipelines”, submitted 2014. Patent Application Docket 37000.00- (KFUPM Ref: NSTIP 861).
2. **Anas Al-Roubaiey**, Tarek Sheltami, and Ashraf Mahmoud, “ARQoS: Adaptive Reliability Protocol for Wireless Sensor Networks” submitted 2014, KFUPM Ref: COE 918, Oblon Ref: 446633US8.
3. **Anas Al-Roubaiey**, Tarek Sheltami, and Ashraf Mahmoud, “Id-based Routing Protocol for Grid topology,” submitted 2014

Journal Papers

1. **Anas Al-Roubaiey**, Tarek R. Sheltami, Ashraf S. Mahmoud, "EATDDS: An Energy Aware TinyDDS protocol for WSN," IEEE Transactions in Parallel and Distributed Systems, to be *submitted 2015*; (ISI)
2. Tarek R. Sheltami, **Anas Al-Roubaiey**, Ashraf S. Mahmoud, "RTDDS: A Reliability Implementation in Wireless Sensor Networks," Sensors, *submitted 2015*; (ISI)
3. Tarek R. Sheltami, **Anas Al-Roubaiey**, Ashraf S. Mahmoud, " A Survey on Implementing Publish/Subscribe Middleware over Wireless Sensor/Actuator Networks," Wireless Networks, *Accepted 2015*; (ISI)
4. Basem Almadani, Mohammed Alsaedi, and **Anas Al-Roubaiey**, "QoS-Aware Scalable Video Streaming Using Data Distribution Service," Multimedia Tools and Applications, *Accepted 2015*. (ISI)
5. Uthman Baroudi, **Anas Al-Roubaiey**, Samir Mekid, Abdelhafid Bouhraoua, and Yau Garba, "Smart Bolts Monitoring Using Wireless Sensor Networks: Implementation and Performance Evaluation," International Journal of Distributed Sensor Networks, 2014. (ISI)
6. Uthman Baroudi, **Anas Al-Roubaiey**, Samir Mekid, Abdelhafid Bouhraoua, "Delay characterization and performance evaluation of cluster-based WSN with different deployment distributions," Future Generation Computer Systems, 2014. IF: 2.6 (ISI)
7. Basem Almadani, **Anas Al-Roubaiey**, and Zubair A. Baig, "Real-Time QoS-Aware Video Streaming: A Comparative and Experimental Study," Advances in Multimedia, 2014.
8. **Anas Al-Roubaiey**, and M. AL-Rhman Alkhiaty, "QoS-Aware Middleware for Ubiquitous Environment: A Review and Proposed Solution," Journal of Computational Engineering, 2014.
9. Basem Almadani, **Anas Al-Roubaiey**, and Rashad Ahmed, "Manufacturing Systems Integration using Real Time QoS-Aware Middleware," Advanced Materials Research, 2013.
10. Basem Al-Madani, **Anas Al-Roubaiey**, Mohammad F. Al-Hammouri, "Performance Enhancement of Limited-Bandwidth Industrial Control Systems," Advanced Materials Research, 2013.
11. Tarek R. Sheltami, **Anas Al-Roubaiey**, Elhadi Shakshuki, Ashraf S. Hasan Mahmoud, "Video transmission enhancement in presence of misbehaving nodes in MANETs," Multimedia Systems, 2009; (ISI)

Conference Papers

1. **Anas Al-Roubaiey**, Tarek sheltami, Ashraf Mahmoud, "A Publish/Subscribe Middleware Cost in Wireless Sensor Networks: a review and case study," to be appear in the proceedings of 28th annual IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'2015).
 2. Uthman Baroudi, **Anas Al-Roubaiey**, "Mobile Radio Frequency Charger for Wireless Sensor Networks in the Smart Grid," The IEEE International Wireless Communications and Mobile Computing Conference (IWCMC 2014) on August 4-8, Cyprus, 2014.
 3. **Anas Al-Roubaiey**, Basher Al-Gohi, "Coverage Optimization of Wireless Sensor Networks with Normal Distribution," Proceedings of the 18th IEEE International Computer Science and Engineering Conference (ICSEC) , Thailand, July 30, 2014.
 4. Basem Al-Madani, Mohammed Al-Saeedi, **Anas Al-Roubaiey**, "Scalable Wireless Video Streaming over Real-Time Publish Subscribe Protocol (RTPS)," IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), 2013, Delft, Netherlands, 2013.
 5. Uthman Baroudi, **Anas Al-Roubaiey**, Samir Mekid, Abdelhafid Bouhraoua, "The Impact of Sensor Node Distribution on Routing Protocols Performance: A Comparative Study," The 11th IEEE International Conference on Ubiquitous Computing and Communications, 2012.
 6. B. Al-madani, **Anas Al-Roubaiey**, T. Al-shehari, "Wireless video streaming over Data Distribution Service middleware," IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS), 2012.
 7. **A. Al-Roubaiey**, T. Sheltami, A. Mahmoud, "Adaptive ACK: A Novel Intrusion Detection System to Mitigate Intended Packet Dropping in MANETs," The International Arab Conference on Information Technology (ACIT), 2010.
 8. **A. Al-Roubaiey**, T. Sheltami, A. Mahmoud, E. Shakshuki, H. Mouftah, "AACK: Adaptive Acknowledgment Intrusion Detection for MANET with Node Detection Enhancement," 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), 2010.
-

REFERENCES

Dr. Tarek Sheltami (supervisor)

Associate Professor

Computer Engineering Department

King Fahd University of Petroleum & Minerals

P.O. Box 89, Dhahran 31261, Saudi Arabia

Phone: +966-3-860-4678

Fax: +966-3-860-3059

Email: tarek@kfupm.edu.sa

Homepage: <http://faculty.kfupm.edu.sa/coe/tarek/>

Dr. Ashraf S. Hasan Mahmoud

Associate Professor

Computer Engineering Department

King Fahd University of Petroleum & Minerals

P.O. Box 1585, Dhahran 31261, Saudi Arabia

Phone: +966 3 860 1724

Fax: +966-3-860-3059

Email: ashraf@kfupm.edu.sa

Homepage: <http://faculty.kfupm.edu.sa/coe/ashraf>

Dr. Uthman Abdurrahman Baroudi

Associate Professor

Computer Engineering Department

King Fahd University of Petroleum & Minerals

P.O.Box 1350, Dhahran, 31261, Saudi Arabia

Phone: +966 3 860-4283

Fax: +966 3 860-3059

Email: ubaroudi@kfupm.edu.sa

Web: <http://faculty.kfupm.edu.sa/coe/ubaroudi/>

Dr. Basem Almadani

Assistant Professor

Computer Engineering Department

King Fahd University of Petroleum & Minerals

P.O.Box 1195, Dhahran, 31261, Saudi Arabia

Phone: +966 3 860-7424

Fax: +966 3 860-3059

Email: mbasem@kfupm.edu.sa

Web: <http://faculty.kfupm.edu.sa/coe/mbasem/>