DDS-BASED REAL-TIME QOS'S IMPLEMENTATION OVER WSAN MIDDLEWARE

BY

Samer Khaled Rabah

A Thesis 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

MASTER OF SCIENCE

In

COMPUTER NETWORKS

December, 2016

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Samer Khaled Rabah** 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 **MASTER OF SCIENCE IN COMPUTER NETWORKS.**

[illegible]

Dr. Basem Al-Madani
(Advisor)

[illegible]

Dr. Ahmad Al-Mulhem
Department Chairman

[illegible]

Dr. Tarek Sheltami
(Member)

[illegible]

Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

[illegible]

Dr. Hosam Rowaihy
(Member)

5|3|17

Date

Dedication

I dedicate this work to my parents, my siblings, my family and friends.

Thank you for supporting me along the way.

Without your praying and help, I could not have completed this work.

ACKNOWLEDGMENTS

All thanks and praises be to my lord (ALLAH), the Almighty, for giving me thorough strength, patience and ability to complete this work.

I would like to thank my parents and my siblings for allowing me to realize my own potential. All the prayers and support they have provided me over the years was the greatest gift anyone has ever given me.

I would like to express my gratitude to my research supervisor, Dr. Basem Al-Madani, for his guidance, which helped me to complete different phases of this thesis and its acknowledgment. I would also like to thank my thesis committee i.e. Dr. Tarek Sheltami and Dr. Hosam Rowaihy, for their valuable feedbacks which shaped the final picture of this work.

I also need to thank Dr. Anas Alroubaiy and my friends everywhere for their prayers, support and encouragement all the time.

Last but not the least I would like to thank lecturers and staff of King Fahd University of Petroleum and Mineral for guiding me to accomplish my Master degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Full Name	: Samer Khaled Yousef Rabah
Thesis Title	: DDS based Real-Time QoS's implementation over WSAN Middleware
Major Field	: Computer Networks
Date of Degree	: December 2016

Wireless Sensor and Actor networks (WSANs) proposed itself to be an emerging technology these days. They have been used in several critical fields such as military, healthcare, environment, and industry. WSAN still suffers from well-known challenges that affect its performance with respect to end-to-end delay, throughput, bandwidth and other resources' utilization. Computer and sensor lab researchers keep developing WSANs to overcome most of these challenges, and other new models have been applied such as a real-time publish/subscribe model for its well-suited characteristics. This model interacts as a middleware software under application layer that guarantees the quality of services (QoSs) and solves the heterogeneity problem with efficient use of resources. State of the art solutions of Real-time pub/sub based middleware have been developed, such as TinyDDS, which is a lightweight version of the Data Distributive Services (DDS) standard of real-time pub/sub middleware. Although this middleware (TinyDDS) supports DDS standard QoSs, it still lacks the implementation of some of the policies such as Time Based Filter and Deadline QoSs. In this work, these critical Real-time QoS policies were implemented over Broker-less TinyDDS middleware, and then a comparison test and analysis have been done to check the performance cost and improvements over WSANs.

ملخص الرسالة

الاسم الكامل: سامر خالد يوسف رباح

عنوان الرسالة: تنفيذ جودة الخدمة الآنية المستخدمة في المعيار العالمي لنظام توزيع البيانات على البرمجة الوسيطة في شبكات الاستشعار و المحركات اللاسلكية.

التخصص: هندسه شبكات الحاسب الآلي

تاريخ الدرجة العلمية: ديسمبر, 2016

تقدم شبكات الاستشعار والمحركات اللاسلكية نفسها كتوجه ذو اهتمام كبير من قبل الأنظمة التكنولوجية في ايامنا الحالية، حيث تم استخدامها في العديد من التطبيقات المهمة والحرجة مثل حقول التطبيقات العسكرية، والصحية والصناعية. لكنها ما زالت تعاني من مجموعة من التحديات والمشاكل المهمة كمشاكل التأخير في الاتصال والاستخدام الفعال لسعة شبكات الاتصال وضعف مواردها الخاصة.

ولا زال الباحثون في مختبرات المستشعرات والحاسب الالي يواصلون البحث والتجارب التي تساهم في تطوير شبكات الاستشعار والمحركات اللاسلكية حتى تتغلب على هذه المشاكل الهامة، وقد قام العديد من هؤلاء الباحثين بتطوير مجموعة من الأنظمة المعيارية وتطبيقها مثل نظام النشر والاشتراك في الزمن الحقيقي و ذلك لخصائصه الفعالة المناسبة لمثل هذا النوع من الشبكات اللاسلكية.

يتفاعل هذا النموذج او النظام كبيئة برمجية وسيطة تفصل بين تطبيقات الحاسوب و الطبقات التحتية في جهاز الحاسب الالي, حيث تؤمن هذه البرمجية الوسيطة الكفاءة في الخدمة وتحل مشاكل عدم التناسق بين المستشعرات المختلفة مع استخدام فاعل للموارد.
و من هذه البرمجيات الوسيطة المستخدمة في شبكات المستشعرات و المحركات اللاسلكية والذي طرح نفسه مؤخرا هو برمجية TinyDDS وهي نسخة خفيفة واصدار معدل عن المعيار العالمي المطروح من شركة OMG والذي يدعى نظام DDS . في حين ان TinyDDSهو نسخه معدلة من DDS لتناسب المستشعرات اللاسلكية البسيطة الا انها ما زالت تعاني من نقص في بعض سياسات كفاءة الخدمة في الزمن الحقيقي مثل الفرز حسب الوقت ونهاية الموعد.

في هذا العمل تم تطبيق هذه السياسات الخاصة بجودة الخدمة في نسخة TinyDDS (التي لا تعتمد على اجهزه وسيطه) لكي يتم تطويره ومن ثم تم عمل مقارنة للنظام قبل وبعد تطبيق هذه السياسات ومن ثم تم تحليل النتائج و تقييم الاداء الذي نتج عنها.

CHAPTER 1

INTRODUCTION

Wireless sensor networks (WSN) are composed of tens to hundreds of tiny devices which are relatively low cost and limited in their capabilities. They are deployed to an area of interest for monitoring a particular phenomenon behavior. Usually, using a sink node, the sensors collect the data and its flow forwarded to the sink/base station which is connected to a monitoring application, as shown in Figure 1.1. The deployed sensors sense and transmit the data to the sink node using one-to-many communication pattern [1].

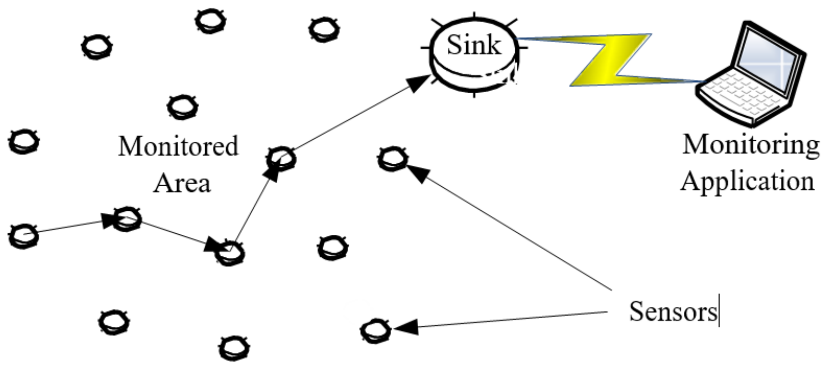


Figure 1.1: Traditional WSN architecture [22]

In Wireless Sensor and Actor Networks (WSANs), the sensors and actors/actuators perform the occurrence of sensing and acting respectively. Some applications use integrated sensor/actor nodes instead of actor nodes who have ability to sense and act, both, at the same time, like in distributed robotic systems.

According to the automation of the WSAN's applications, these applications can be classified into two categories: A) *Partially automated applications,* where the network control is more centralized at the sink or base station, and as a result, this delays the response to the processing results. B) *Fully automated applications*, where the sensors are capable of sensing the data and directly sending this data to the actuators for further processing, as needed, and acting accordingly. The fully automated type reduces the overall response time and overhead, which is more suitable for real-time applications [2]. Figure 1.2 shows the WSAN architecture for both types.

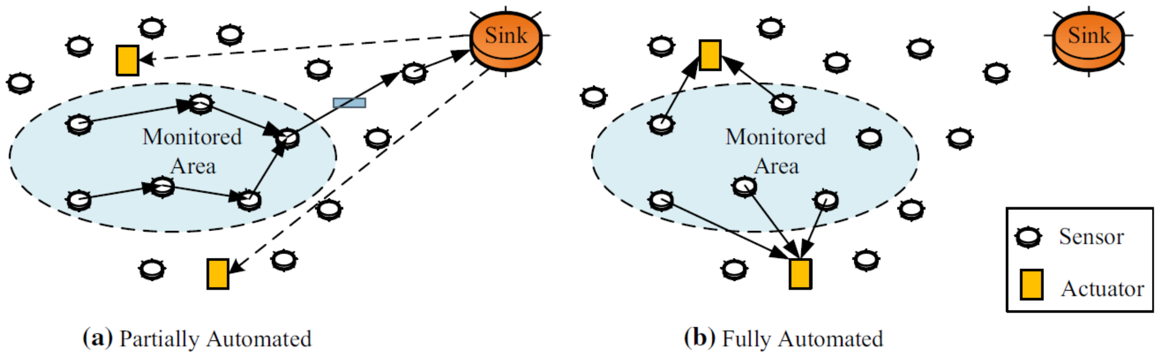


Figure 1.2: WSAN architecture with partially and fully automated interaction [22]

The publish/subscribe scheme is a messaging-based communication model which is supported by many industrial and research prototypes. In this model, with less information about the receiver and its address, the publishers (senders) send their data to a logical space, called middleware. Similarly, with less information about the sender and its address, the subscribers (receivers) receive only the data in which they are interested. Since pub/sub scheme strength lies in the full decoupling in time, space, and synchronization between publishers and subscribers, it is proposed as a suitable solution for large-scale distributed real-time applications [4].

Enabling publish/subscribe model in WSAN would be a key solution to overcome many of its problems. Moreover, it improves the WSAN's performance by providing great advantages such as easy development of applications, portability, scalability, real-time properties and QoS support. A suitability analysis that study the suitability of publish/subscribe scheme for WSAN is mentioned in [22] as follows:

- Pub/sub model has scalability advantage in term of deployment and message delivery in WSANs that have a large number of sensors, actors, and sinks.
- Pub/sub model is an event-based scheme which is suitable for frequent data updates in monitoring and control systems.
- Pub/sub model is suitable for a high degree of common interest in applications, sensors, sinks and actors.
- Pub/sub model is suitable more than request/reply model for less user intervention applications.
- Pub/sub model is a real-time model that guarantees an immediate data update and delivery to the subscriber of the short deadline.
- Published/subscribed model is not suitable if clients rarely use published data.

The main characteristics and issues that introduce the publish/subscribe model as a suitable solution for WSANs are [22] summarized as follows:

1- **Many-to-Many Interaction**: Since WSANs of multiple base stations and sinks migrate the applications from one-to-many to many-to-many communication model, the pub/sub model is suitable in this case. The data is supposed to move in both directions from sensors as publishers to sinks or base stations as subscribers, and vice versa; from the base stations or sinks as publishers to the actuators as subscribers for some reaction.
2- **Data-Centric**: this is one of the WSAN's key features that distinguishes them from other wireless networks, where they are not interested in the nodes' identity but in the data that is being transmitted. As a result, this requirement is satisfied by using the data-centric publisher/subscriber communication model, where the subscribers are interested in the information received from the publishers but not from their addresses.
3- **Network Dynamics**: since the sensor nodes are joining and leaving the WSANs in a dynamic manner due to hardware failures or energy exhaustion, publish/subscribe interactions model is the suitable solution where it hides the underlying details from WSAN's applications in order to mitigate the continuous addressing change due to joining to or leaving from the network.
4- **Heterogeneity**: a complex and expensive process is required to develop an operating system that is capable to connect heterogeneous systems. The Pub/sub middleware, due to intensive efforts by researchers and developers, comes to mitigate the problems of connecting different nodes' platforms. The pub/sub middleware as an intermediate layer between the underlying platforms and the applications, facilitates the development,

portability, and interoperability. Figure 1.3 shows the effect of middleware to hide the underlying layers complexity.

The pub/sub middleware is proposed to be a well-suited solution to develop the WSAN's applications. Even so, several challenges would face the developers to adapt the pub/sub middleware to meet the requirements of WSANs and QoSs needed. This issue attracts researchers' attention to propose a pub/sub middleware for WSANs, some of the state of the art solutions are Directed diffusion, Mires, TinyCOPS, MQTT-S, TinyDDS, UPSWSN-MM, and PS-QUASAR.

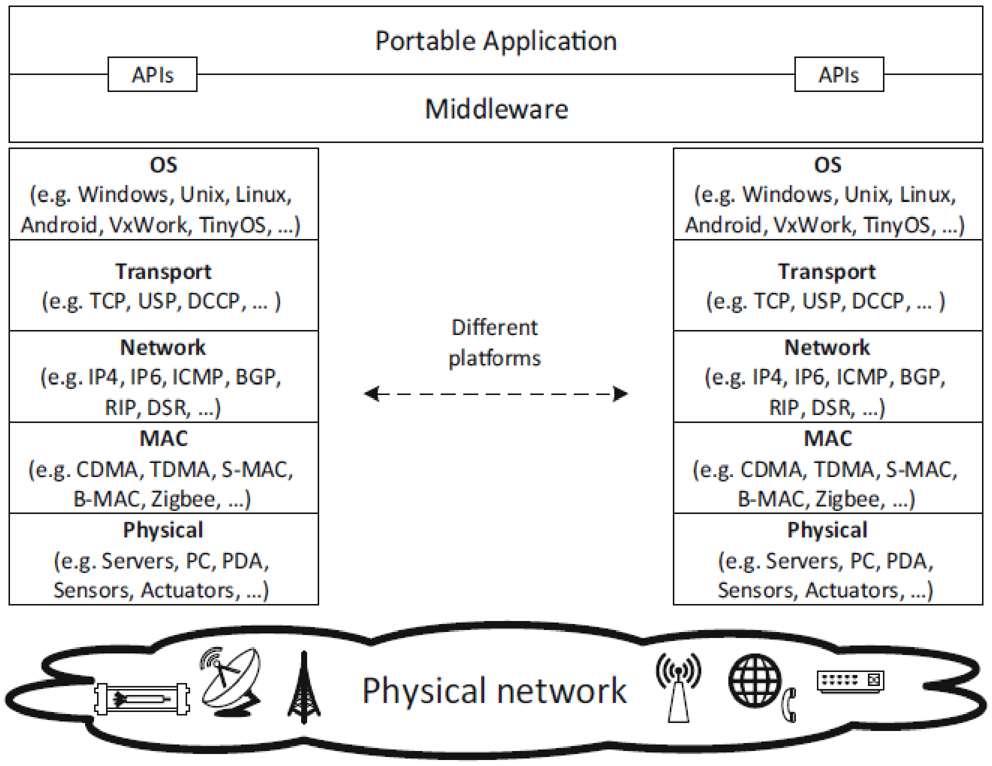

Portable Application
APIs
APIs
Middleware
OS
(e.g. Windows, Unix, Linux, Android, VxWork, TinyOS, ...)
Transport
(e.g. TCP, USP, DCCP, ...)
Network
(e.g. IP4, IP6, ICMP, BGP, RIP, DSR, ...)
MAC
(e.g. CDMA, TDMA, S-MAC, B-MAC, Zigbee, ...)
Physical
(e.g. Servers, PC, PDA, Sensors, Actuators, ...)
Different platforms
OS
(e.g. Windows, Unix, Linux, Android, VxWork, TinyOS, ...)
Transport
(e.g. TCP, USP, DCCP, ...)
Network
(e.g. IP4, IP6, ICMP, BGP, RIP, DSR, ...)
MAC
(e.g. CDMA, TDMA, S-MAC, B-MAC, Zigbee, ...)
Physical
(e.g. Servers, PC, PDA, Sensors, Actuators, ...)
Physical network

Figure 1.3: Middleware layer hides the complexity of underlying layers [22]

1.1 Background and Terminology

In this section, as a pub/sub middleware standard, Data Distribution Service standard and its quality of services will be described briefly.

1.1.1 Data Distribution Service

Data Distribution Service (DDS) standard is a real-time middleware, developed by Object Management Group (OMG) based on pub/sub model. Since OMG is an object-oriented developer in software technology, it aims to add portability, interoperability, and reusability features to highlight its object-based software to be applicable for distributed heterogeneous environments [28].

DDS pub/sub model, used to be a powerful method of information dissemination that links anonymous data publishers to data subscribers. One-to-many and many-to-many distribution mechanisms are both available in DDS which allow distributing data between individual publishers and subscribers or group of large numbers of both, this flexibility is free from publishers and subscribers places and addresses information.

For writing and reading data in DDS two abstractions were provided: Data Reader (DR), and Data Writer (DW) [19]. Figure 1.4 and Figure 1.5 show DDS Distribution model and pub/sub model respectively:

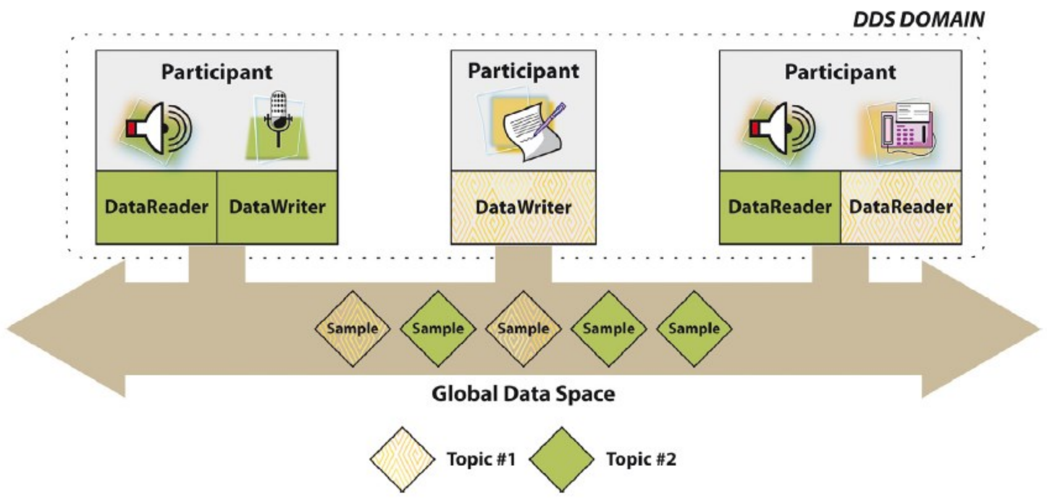


Figure 1.4: Distribution model for DDS [25].

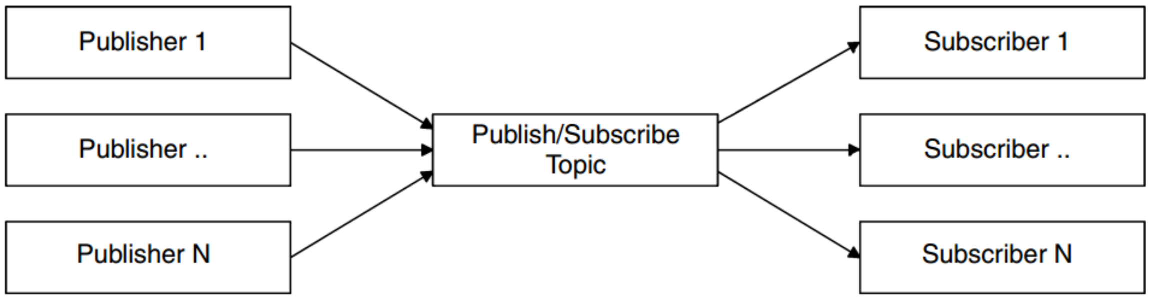


Figure 1.5: DDS pub/sub model

DDS was basically designed as a result of many types of research over the difficulties, the real-time applications, may suffer such as immediate data sending from the source publishers to the destination subscribers directly without the need of brokers (intermediate servers).

In DDS pub/sub application composed of participants which can be a publisher, subscriber, or publisher/subscriber at the same time. Each of these is running on a separated different address machines and simultaneously publish and subscribe to a Topic of data streams identified by unique topic names which compromise the data type, and data associated QoS.

Since that scalability is one of the features of this model, some keys can be used within topics, this allows to receive the data from hundreds of similar data streams with a single subscription. Also, these keys are used by middleware engine for efficient processing of sorting and delivering [18] [19].

Several implementations on DDS take place in research and industry which can be categorized into free (open source) such as Open Splice and Open DDS, and commercial, such as CoreDX and RTI-DDS [29].

1.1.2DDS Quality of Service Policies (QoS's)

DDS had a great advantage over real-time Quality of Service (QoS) controlling. Since QoS is a set of characteristics that drives the behavior of the service, DDS relies on the application requirements to determine the QoS's and each pair of (a publisher and a subscriber) participant can establish its own QoS's agreements.

Since the QoS parameters are implemented as a contract between the participants (Publisher offers), (subscriber requests), and (levels of service), it becomes the middleware responsibility to match the offers and requests, before establishing the connection or incompatibility error will be shown.

Here are some examples of usual used QoS in DDS [18]:

- Reliability: This QoS determines the level of reliability requested by the subscriber or offered by the publisher.

- Durability: This QoS allows an application to send data even if there are no current subscribers on the network.
- Time Based Filter: States that the subscriber doesn't want to receive more than one value each minimum separation of time from a subset of values, this would be critical in WSN due to limited resources of data rate and processing time; therefore Time Based Filter expresses the data rate threshold which the subscriber can handle.
- Deadline: This QoS controls the maximum time to send and receive topic samples and it's the middleware responsibility to supervise the instances updating rate between both DW and DR sides. For consistency, the deadline time period should be greater than Time Based Filter.
- Transport Priority: This QoS is to allow the application to take advantage of transports capability of messages sending in different priority specifications.

1.2 Problem Statement& Contributions

Real-time WSAN's applications may encounter some challenging problems such as latency and data loss that occur due to congestion, bandwidth limitations, and limited hardware recourses. These serious issues will decrease the network overall throughput and shortage the lifetime of nodes in term of power [1], therefore the researchers were motivated to find out suitable models to address these challenges [3].

State-of-art solutions of Real-time WSAN’s pub/sub middleware were proposed such as TinyDDS, and PS-QUASAR [23]. However, TinyDDS middleware is superior, since it is a lightweight version based on the OMG standard DDS. It is still in the development stage and many of Real-time DDS critical QoS’s suitable for WSAN’s such as Time based Filtering, and Deadline are not yet implemented in the middleware [23], which will improve the sensor networks performance and overcome the limited resources problem [26].

Since TinyDDS middleware lacks the implementations of Time based Filtering, and Deadline, in this work these critical policies were added and implemented to upgrade the middleware, after that its performance has been evaluated and tested in such a comparison before and after implementing this quality of services policies.

2.3 Thesis organization

The rest of the thesis is organized as follows. In Chapter 2, a comprehensive study was provided to several types of research found in the literature that addressing the problems and challenging criteria in WSANs, and some solutions proposed to solve these problems, also it presents the using of pub/sub middlewares as a superior solution to overcome the challenges; specially TinyDDS middleware. Next, the methodology is described in chapter 3. The implementation design of Deadline and Time Based Filter quality of services are described in Chapter 4. In chapter 5, the simulation setup, tools and network topologies were discussed in addition to the performance evaluations that used for testing and comparison before and after implementing the QoS's. In chapter 6 the Conclusions and future directions for the work were presented.

CHAPTER2

LITERATURE REVIEW

Wireless sensor and actor networks (WSAN’s) introduce itself as an emerging revolutionary technology that affects all aspects of our lives. Its great use in multiple applications such as military, healthcare, biological, environmental, structural health and condition based monitoring, forces the researchers in the field of embedded computer and sensor technology to develop it in advance and overcome the critical issues and challenges there. Several studies are addressing most of WSAN’s design influencing factors, take in concern its limited resources and quality of services, to improve it in both level of hardware and software [1] [2] [3].

Since there are many publications in this field, in this section, I will mention the most related ones to my work.

In [4] the pub/sub scheme is introduced, since it is an event-based interaction its strength lies in the full decoupling, in time, space, and synchronization between publishers and subscribers, which is required in large scale settings such as WSAN's.

In [5-12] [24-27] the publishers were focusing on QoS provisioning in WSAN's. These studies may be classified into two approaches: pub/sub based and not- pub/sub based, both examining the QoS's supported in WSAN's and its requirements, which differ depending on the application, also the open research issues in QoS and its critical challenges were discussed.

In [13] a new operating system platform specifically designed for WSAN's called TinyOS was introduced, it is implemented in the NesC language, it combines the limited resources of flexible components with a model execution to support complex concurrent operations, therefore it facilitates the experience on WSAN's, thus it had been used in several researches and developments.

Sensors and actors in WSAN's are different in terms of hardware platforms, which make it clearly impossible to develop an Operating System (OS) that runs on all of them. Therefore, a need to decouple the OS from the hardware platform becomes necessary using middleware which hides the underlying platform differences, and facilitates scalability, interoperability, deployment, and development of the applications [14].

Numerous works on middleware for handheld devices for different operating system have been developed, and many surveys take place in literature to compare between these different middlewares [14-19].

In [14] the publishers illustrate that a huge amount of work the middleware needs before it became suitable for WSAN's due to resource constraint unreliability QoS support and diversity in the sensor/actor hardware, some features and challenges are presented in details and compared for various middleware such as Impala, Mate, TinyDB, Agilla, TinyCubu and TinyLime. However most of these middleware address some of these features, there are still some critical features like security and QoS support which are ignored by most of the middleware.

In [15][16][17] publishers try to show the current state of studies and researches in WSAN's middleware domain. They discussed some features and compare between several middleware such as Mate, Magnet, Cougar, SINA, DsWare, Impala, Milan and Envirotrack. Where these approaches classified into four categories: virtual machine, database based, modular programming, and application driven, most of these middlewares assume that sensor nodes are homogeneous, however not all features and challenges are supported by these middlewares, and still a long way for a perfect middleware for WSAN's.

Data Distribution Service (DDS) is a well-known standard middleware in research and industry for supporting real-time distributed systems based on the real-time pub/sub model. The DDS specification offers several QoS like Reliability, Durability, Resource Limits, Deadline, Time Based Filter, and Transport Priority, also RTI connext DDS is an industrial platform for DDS [18][19]. The DDS standard-based proposed solutions for WSAN's middleware are TinyDDS [20] and μDDS [21], however, TinyDDS is more popular and cited by the majority of researchers in the research community, and also it is an open source.

In [22] a comprehensive review and study for state of the art solutions of publish and subscribe WSAN's middleware such as: Directed diffusion (2003), Mires (2005), Quad-PubSub (2007), TinyCOPS (2008), MQTT-S (2008), TinyDDS (2009), MiSense (2009), PUB-2-SUB+ (2010), TinyMQ (2011), UPSWSN-MM (2012) and PS-QUASAR (2013). A comparison had been done between these solutions in terms of features, architectures limitations and QoS mechanisms they supported related to Reliability, Priority, Deadline, and Energy-awareness. The reviewers mentioned that there is still a need for more effort in design and implementation, in addition to that these solutions lack efficient ways to deal with performance factors like churn and failure rates and energy-aware dynamic load distribution on the network. TinyDDS and PS-QUASAR were superior over other solutions.

However TinyDDS [20] and PS-QUASAR [12] propose themselves as super state of the art solution for WSAN's middleware [23]. TinyDDS is a lightweight version of DDS standard for embedded systems that is standardized by the Object Management Group (OMG) organization in 2003. It has several potential enhancements that can significantly reduce the overhead, such as using broker less architecture, its integration with the enterprise networks becomes straightforward, also supports QoS for WSAN's. Hence, that TinyDDS supports QoS's there is no implementation of these QoS's yet.

In [22] the main methods of routing for both types of messages (subscription or data) are either broker-based or broker-less. Since TinyDDS uses the broker-based methods in routing; this centralized method is not suitable for WSANs functions and platforms, thus it causes a bottleneck which consumes the node energy rapidly, so ends the network lifetime in short period. Therefore, it's better to use Broker-Less TinyDDS (BLTDDS) which assumes that the middleware has a previous knowledge about all the publishers in the networks since the time of deployment, so that all subscribers broadcast subscription messages to all nodes in the network, then the matching process will be in publisher side.

2.1 Pub/sub (pub/sub) Model

2.1.1Pub/sub Components

The pub/sub model was developed for the benefits of scalability, flexibility, and fast data delivery, therefore it has been proposed as main solution for large-scale distributed systems [22]. Figure 2.1 explains the main components of pub\sub model and its basic model [22].

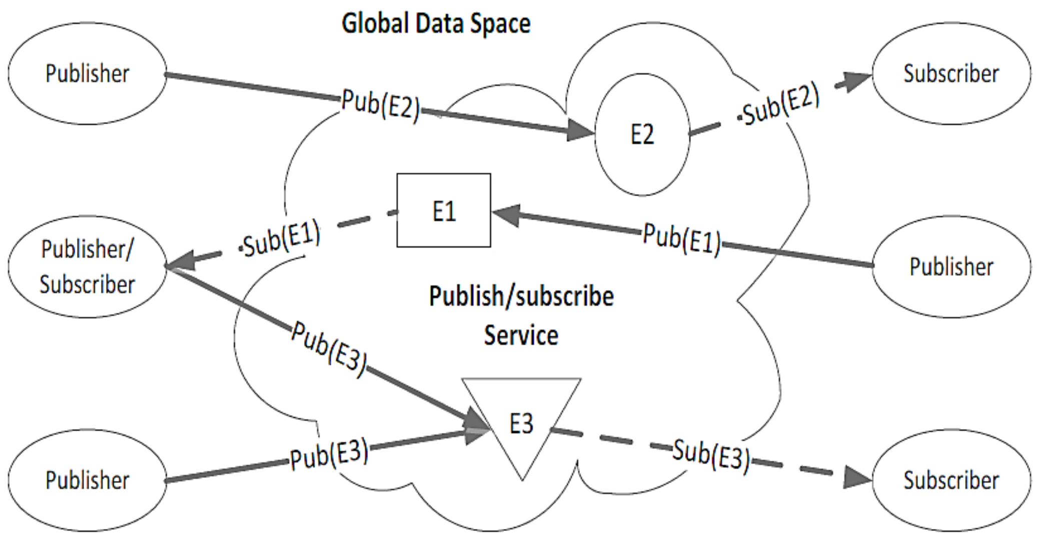


Figure 2.1: The core component of pub/sub model [22].

The main component of pub/sub scheme is notification service (pub/sub service) which basically provides and manages the storage service and subscriptions. As the figure above illustrates that the global data space represents the real implementation of the distribution over brokers (servers) and the end-nodes in the system [22].

The notification service playing the role of moderating and matching between publisher and subscribers. The subscriber for specific events, i.e. E1, E3, using subscribing function sub (E) to subscribe, then the notification service matches it to the right events of the publishers, and it completes the data delivery to the subscriber. These processes in the system classified to Three main operations: pub (E) function to publish the events, sub (E) function to subscribe to a specific event, and the unsubscribe function. The participants are either a publisher or a subscriber or both at the same time [22].

Since that notification service (pub/sub service) provides scalability and flexibility, this happened in three dimension of decoupling between subscribers and publishers as follow [22]:

- Space: the publishers and subscribers don't need to know each other's to interact where the main interest is the event itself regardless from where it comes or where it goes.
- Time: Especially for the high dynamic network which suffers a high rate of nodes fail or disconnections, the publishers and the subscribers can interact independently at any time.
- Synchronization: asynchronous communication paradigm was used which means no blocking on concurrent tasks of receiving and sending in both sides of subscribers and publishers.

Distributed systems such as WSANs and mobile networks are naturally asynchronous, thus removing dependencies leads to the faster decoupling between the participants and increases the scalability of these systems.

2.1.2 Pub/sub as A Middleware

The Pub/Sub middleware basically consists of five components: end nodes (subscribers or publishers or both), subscription or publishing messages, notification service, Application Programming Interfaces (API) and programming abstraction, and QoS mechanisms that the pub/sub applications support [22]. Figure 2.2 shows the main components of pub/sub middleware:

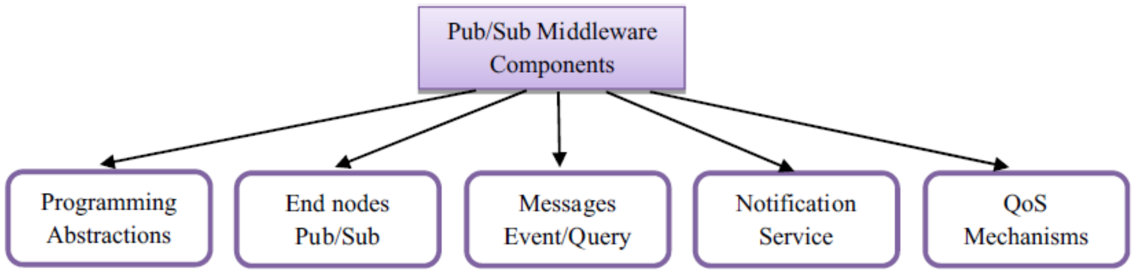


Figure 2.2: The pub/sub Middleware components [22].

Programming Abstractions:

Application Programming Interfaces (APIs) and its abstraction are improving the developing of WSAN application and reduce its complexity. In pub/sub middleware, APIs are used to create, publish, subscribe, and unsubscribe a certain event. This will make the application development easy, and hide (underlying) the details and heterogeneous complexity under the network layers from developers [22].

End-Nodes:

As much as communication systems, the end users in WASN pub/sub middleware nodes are called publishers (senders) and the subscribers (receivers). The publisher creates the events and sends them to the notification service which in turn delivers it to the interested subscriber. In case there is no subscriber dedicated to that event it will be kept in the notification service until either a new subscriber to that event or it reaches its expired time. The subscriber creates an event subscription, then the notification service triggers a matching process if a matching published event is available. If not, the subscription will be kept in the notification service until it matches a published event, or it reaches its expired time [22].

Messages (Event/Query):

There are three different types of messaging in pub/sub middleware interaction paradigm: the advertise message, the event (publication) message or data message, and the query (subscription) message [22].

The advertise messages are used for an event advertisement before publication. These messages, are created by the application, include two parts the header and payload.

The header main fields are identifier, issuer, and some fields dedicated to QoS's parameters. Figure 2.3 illustrates the general message format used in pub/sub WSAN middleware [22].

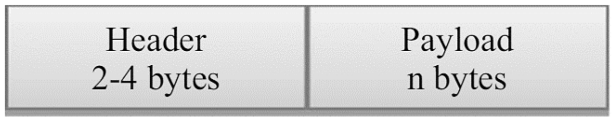


Figure 2.3: The generalmessage format [22].

The query message is sent by subscribers to register all events or part of it, and it is supposed to be important since it can be used to classify the most used Pub/Sub systems. Thus that the subscriber's ways of registering to the events are different because it depends on the implementation. Therefore it affects the architecture used to implement the notification service.

Notification Service (NS):

Notification services are responsible for spreading and expanding the data in pub/sub systems. It mediates between publishers and subscribers, thus it is the heart of the middleware. It has a specific operations to interact with the publishers and subscribers that are illustrated in Figure 2.4, where the publisher issues "publish ()" and "advertise ()", for publishing and advertising new topics; also the subscriber issues "subscribe ()" and "unsubscribe ()" to subscribe and unsubscribe to a topic, in addition "notify ()" can be used to notify the subscriber about matched topic. NS services also include discovering the participants (the publishers and the subscribers), storing the publications and subscriptions, match between them, events routing, filtering, and managing the pub/sub Quality of Services (QoS's).

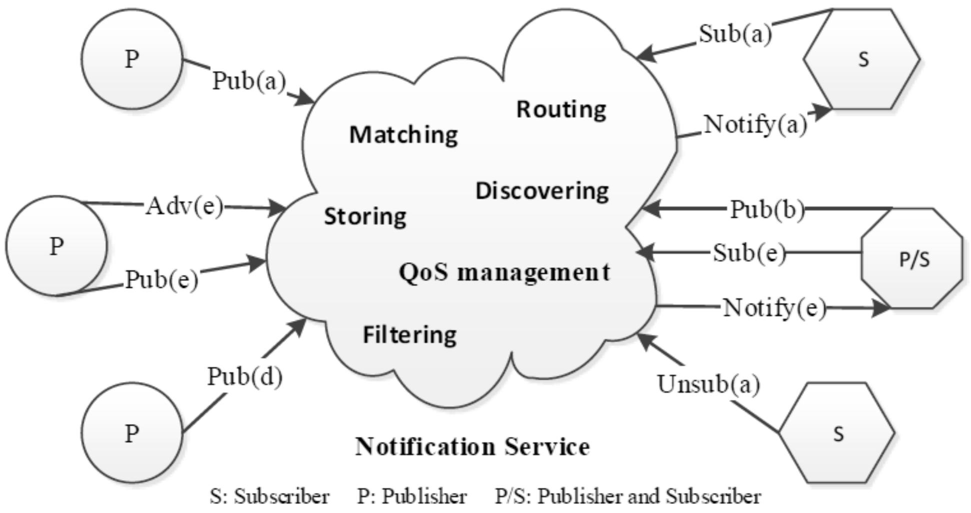


Figure 2.4: Notification Service Operations[22].

Quality of Service Mechanisms (QoS's):

Quality of Service (QoS) is considered as advanced features for any WSAN middleware. Since the behavior in pub/sub systems is less deterministic because of decoupling principle, thus make it neither simple nor an easy task to support QoS's, especially in resource-limit constraint systems. Middleware is responsible for guaranteeing the QoS's after negotiation if the application layer QoS's requirements cannot be satisfied by the network under layers [22].

2.2 Pub/sub in WSANs

In this section, some pub/sub based solutions in the past years for WSN/WSAN will be discussed as a comparative study. Then pub/sub WSAN general middleware reference model will be presented in the end.

2.2.1Pub/sub Solutions

Directed Diffusion:

Is considered the earliest pub/sub paradigm for WSANs. It is based on data-centric protocol. The interests (subscriptions) are broadcasted over all network, in meanwhile the gradients should be setup for later use of events drawing (data request). The matching process is done locally be each node after interest examination. If it has the requested data, then the node sends the information to the sink using the interest reverse path. Otherwise, the interest is just propagated throughout the network. No need for brokers which avoids centralized processing disadvantages. However, it has a memory overhead in communications and processing where all nodes do the same for each interest. Cached data can be used in intermediate nodes, also data aggregation thus consumes less energy and minimize the traffic. Data filtering can be achieved using the attribute value feature in the data structure. For each received interest, it has its own gradient towards the node sending the interest. A secured version was proposed recently that improves the integrity and data authenticity with low overhead [22].

PS-QUASAR

It is a pub/sub middleware solution where all nodes in the network are publishers for each topic. It provides high programming level and QoS support such as reliability and priority. It handles a many-to-many messaging exchange by means of multicasting techniques. It consists of three modules: API, routing module, and maintenance protocol, Figure 2.5 depicts the architecture and the interconnection between modules are interconnected.

The maintenance protocol discovers the pub/sub terminals (publishers/subscribers) and creates the links between neighbor nodes. Routing module collects the information from maintenance protocol to be used in events routing. Since topic-based has less matching overhead than content-based, the API module use it in developing WSANs applications in this middleware.

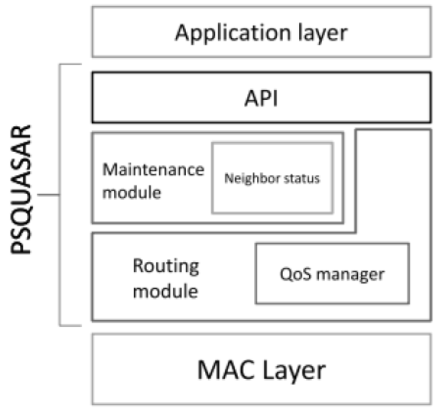


Figure 2.5: PS-QUASAR Architecture[22].

Tree routing protocol is used as an enhancement developed from Bellman-Ford algorithm. Despite that PS-QUASAR is energy efficient, QoS's aware, and using a robust routing protocol, it suffers some critical issues such as memory space limitations, Also, the deterministic behavior of nodes deployments in WSANs is just evaluated in term of performance [22].

TinyDDS:

It is an OMG DDS standard adopted for WSAN's. It is a lightweight pub/sub middleware that allows the applications to bypass over the boundary of WSAN's and provides them an access to the networks, regardless of their protocols, platforms and programming languages they use.

In addition to that, it allows the WSAN's applications to have a powerful control over nonfunctional properties of the middleware level and the application level, and further specialized in their own requirements flexibly. It can automatically address the dynamic network behaviors and conditions, which according to that performs an adaptive event publication and balances its performance regarding conflicting objectives using an evolutionary optimization mechanism of the multi-objective.

TinyDDS main contributions for WSAN's are providing interoperability for accessing the networks, also the flexibility of customizing nonfunctional properties such as event filtering, data aggregation and routing [20]. TinyDDS despite of its great services for WSAN's, it still needs more developmet since it lacks the energy-aware support and the QoS's features not yet implemented to handle the limited resources of WSAN's [23]. Figure 2.6 describes the TinyDDS architecture and its main components for MicaZ platform.

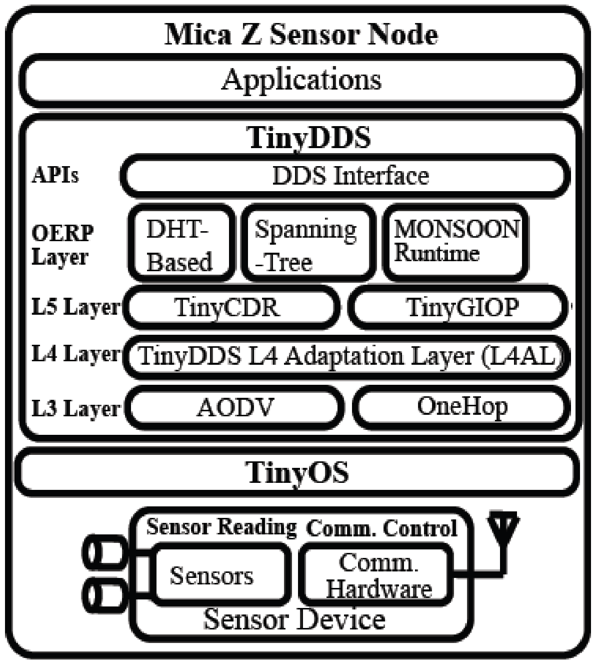


Figure 2.6: TinyDDS architecture over TinyOS and MicaZ platform [22]

2.2.1: WSAN Pub/Sub Reference Model

This model is proposed as a reference model for pub/sub middleware for WSANs. It had been extracted by [22] after full survey for all available pub/sub solution's architectures. The general case for middleware layer is to be between the application and the operating systems layers.

The pub/sub middleware will be considered a complete solution if it consists of four main components that mentioned before in middleware components in section 2.1.2, add to it the messaging component. Different implementations may use different services and QoS's. However, in WSAN platforms it is very critical to add these feature due to their resource constraints.

Thus it's a challenging issue for the middleware design where it significantly depends on the application requirements in WSANs. TinyOS and Contiki are the most used platform operating systems. Figure 2.7 shows the general Middleware reference model architecture.

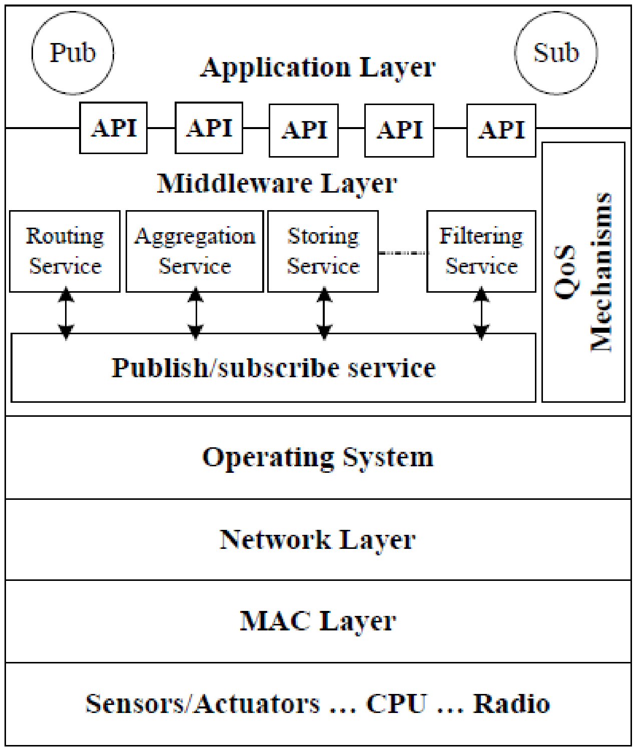


Figure 2.7: General Middleware reference model architecture [22]

Table 2.1 and Table 2.2 compares the proposed prototypes for pub/sub model and it summarizes the implementation and evaluation issues of each proposed solution in literature. Table 2.3 summarizes the features and limitations to the mostly used simulators in the literature for evaluating pub/sub solutions of WSN/WSAN.

Table 2.1: Pub/Sub WSAN Solutions (where; D: Deadline; P: Priority; R: Reliability)[22]

Solution	**Sub Scheme**	**Overlay Infrastructure**	**Multiple Sinks**	**Actuator Support**	**QoS**			**Energy Awareness**	**Mobility**
					Reliability	**Priority**	**Deadline**		
Directed Diffusion (2003)	Topic/ content based	P2P	Y	N	N	N	N	Y	Y
Mires (2005)	Topic based	P2P	N	N	N	N	N	N	N
TinyCOPS (2008)	Content based	Broker/P2P	Y	N	N	N	N	Y	Y
MQTT-S (2008)	Topic based	Broker	N	Y	Y	N	N	N	N
TinyDDS (2009)	Topic/ Content based	P2P	Y	N	Y	Y	Y	N	N
UPSWSN-MM (2012)	Content based	Broker	Y	N	Y	N	N	N	Y
PS-QUASAR (2013)	Topic based	P2P	Y	Y	Y	Y	Y	Y	N

Table 2.2: Pub/Sub WSAN Solutions Evolution and Features Summary [22]

Solution	Testapproach	Testing tools	Performancemeasurements	Remarks
Directed Diffusion	analytical/simulation	NS2	Avg.dissipatedEnergy/Avg. delay/distinct-eventdelivery ratio	Dataaggregation,reversepath reinforcement,analyticalanalysis for datadeliverycost,distributed matchingprocess
Mires	none	None	CaseStudy Anenvironmentmonitoring Apps /nomeasurements	Dataaggregation,Topic advertisement,focusedonfacilitating WSNappsdevelopment
Quad-PubSub	simulation	JiST/SWANS	Msgs overhead/event;Hopsvs subscribers;	Supportfor resource-awareness and sharedeventsdisseminationpaths
TinyCOPS	Indoortestbed	TWIST/TinyOS	Subscriptionsandnotifications deliveryratio/activepublishers /PSLOC*/flashandRAMsize	Themainproperties are the decouplingofcommunication protocols andtheadaptivematching point
MQTT-S	testbed	TinyOS; TmoteandMicaZ	Just SAmemoryfootprint (12Bytes)	Seamless integrationoftheWSNwith traditional Networks (MQTTbased)
TinyDDS	simulation/ testbed	TinyOS, TOSSIM, / SunSPOT, Solarium emulator.	PKTheaderoverhead;Memory Footprint; Processing;and powerconsumption.	Standard-basedsolution (OMGDDS); seamless integrationwithaccess networks.
PUB-2-SUB$^+$	simulation	Ownsimulator	No.ofhopspereven/query;No. of replicas perquery; Notificationdelay;storage, comm.,computationloads.	Content/basedrouting;noneedfor locationinformation;less overhead thangossiprouting;
TinyMQ	simulation	OPNET	Comparisonwithpub-2-subin hops/queryandnotification delay;andrepaircost(number of repairednodes)	AddinginteroperabilitywithinWSN; content-basedroutingwithoutlocation information.
UPSWSN-MM	Outdoortestbed	HTCsmartphones with AndroidOS;T mote sensors withContiki OS;Apacheser	Delay;numberofdelivered data;communicationoverhead	Supportinginternetusers toget sensingdata anytimefrom anywhere; integrateWSNtointernetviamobile phones.
PS-QUASAR	Simulation	Contiki(OS)Telos B motes;Coojasimulat	Energyconsumption;delivery ratioofpackets;delay	QoS supportandhighlevel programming;multicastsupport

Table 2.3: Simulators Used in Evaluating Pub/Sub Solutions for WSAN [22]

Simulator	**Language**	**GUI**	**Generality**	**Open Source**	**License**	**Features**	**Limitations**
TOSSIM	nesC	No	WSN	Yes	Free	*Apps porteddirectlyto HW platform. *Bit-levelsimulation	*Restricted forTinyOS. *Lackdecent documentations. *Add-ontosupportenergy consumption,*PowerTossim* z[110]
COOJA	Java/C	Yes	WSN	Yes	Free	*BestchoiceforContiki-basedWSN *Abletosimulatenon-Contiki nodes *easytouse andunderstand *Supportlarge-scaleprotocols andalgorithms	*Supports alimitednumber of Simultaneousnodetypes. *Makingextensiveandtime dependentsimulations difficult.
OPNET	C++	Yes	General	Only protocol models sources	Comme-rcial	*Lots ofprotocolmodels includingTCP/IP,ATM, Ethernet,etc. *SimpleGUItobuilddifficult scenariosandgetsimulation results.	*Expensive *quite difficulttomodify theprotocols
NS-3	C++	No	General	Yes	Free	* Supportreal-time scheduling,multipleradio interfaces,andmultiple channels. * Packet-levelsimulation.	*Lackofanapplication model. *Codenotportableto HW. *NotscalableforWSN.
GloMoSim	C/Parsec	Yes	General	Yes	Free	*supports purelyforwireless networks protocols. *UsingstandardAPIs betweensimulationlayers. *parallelsimulationsupport	*Less accurateinsensor networks simulations. *Codenotportableto HW.
Castalia(basedon OMNET++)	C++	Yes	General	Yes	Free	*HighlytunableMAC protocolandaflexible parametric physicalprocess model. *Applicationlevelsimulator	*Notasensorspecific platform. *Notuseful forportable sensorcode.
PSLOC:Physical SourceLines OfCode							

CHAPTER 3

THE METHODOLOGY& PROPOSED APPROACH

Since WSANs function depends on wireless channels, the centralized method is not suitable to be used in most of its applications. And since we argue that TinyDDS was proposed as the state of the art middleware solution for WSANs' applications, it still uses a broker-based method to deliver the middleware messages between subscribers and publishers. Therefore, in [23] they proposed an improvement to the default TinyDDS and they presented an enhanced version called Broker-Less TinyDDS (BLTDDS), where the usage of brokers is not applied anymore.

In this chapter, Broker-Less TinyDDS (BLTDDS) will be discussed in details in the first section. After that, some of DDS-based real-time QoS policies, which will be implemented as a contribution to improve BLTDDS, will be comprehensively discussed.

3.1 Broker-Less TinyDDS (BLTDDS)

3.1.1Messaging & data delivery

In any pub/sub system, there are two basic phases: the first one is the Discovery phase where any node starts to send subscription or publication messages as soon as it joins the network, thus it can be recognized as either subscriber or publisher. The second phase is the Data dissemination phase where the middleware starts to deliver the interested data to the subscriber from the publishers [23].

According to the routing method, which the middleware uses to deliver the data and messages throughout the network, the middleware can be classified into either broker-based method or broker-less method. The Default TinyDDS (DefTDDS) uses the broker-based routing method t/o route pub/sub messages, where for each topic one broker node is assigned [23].

In Discovery phase, the publishers and subscribers use a hashing algorithm to obtain the broker node address based on the topic identification and max topic numbers. This information is already known to the end-nodes since the network deployment. Then the broker node retrieves all subscription and publication messages from all end-nodes and store them in a list [23].

In Data Dissemination phase, since tiny devices such as sensor or actuator nodes are suffering from memory limitations, the broker node has a volatile memory. Thereby, the published data in dissemination phase is directly deleted from the broker node database list after delivering it to all subscribers. Multicast messaging is used to deliver one publication data message to more than one subscriber if exist. Figure 3.1 illustrates the two phases' processes for DefTDDS [23].

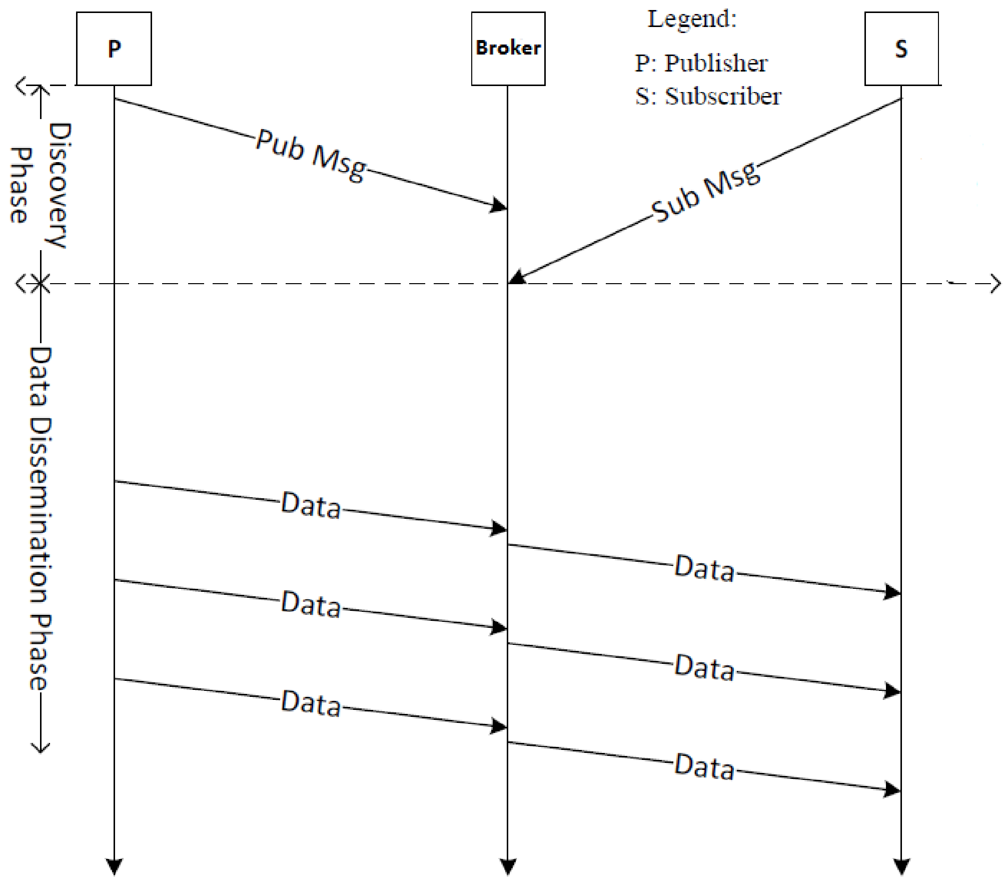


Figure 3.1: Discovery phase and Data Dissemination phase for Default TinyDDS [23]

In contrast, in Broker-less TinyDDS, the broker nodes are eliminated and the two main phases' functionality (Discovery phase and Data Dissemination phase) is distributed by the middleware over the end-nodes (publishers/subscribers). The subscription messages are *broadcasted* from the subscriber to all publishers throughout the network in Discovery phase, and then the publishers decide whether to send to that subscriber or not based on the output of the matching process for the topic and QoS policies. When there is a match, the publishers begin to send the data to that subscriber in the Data Dissemination phase. Figure 3.2 shows the diagram of Discovery phase and Data Dissemination phase for BLTDDS [23].

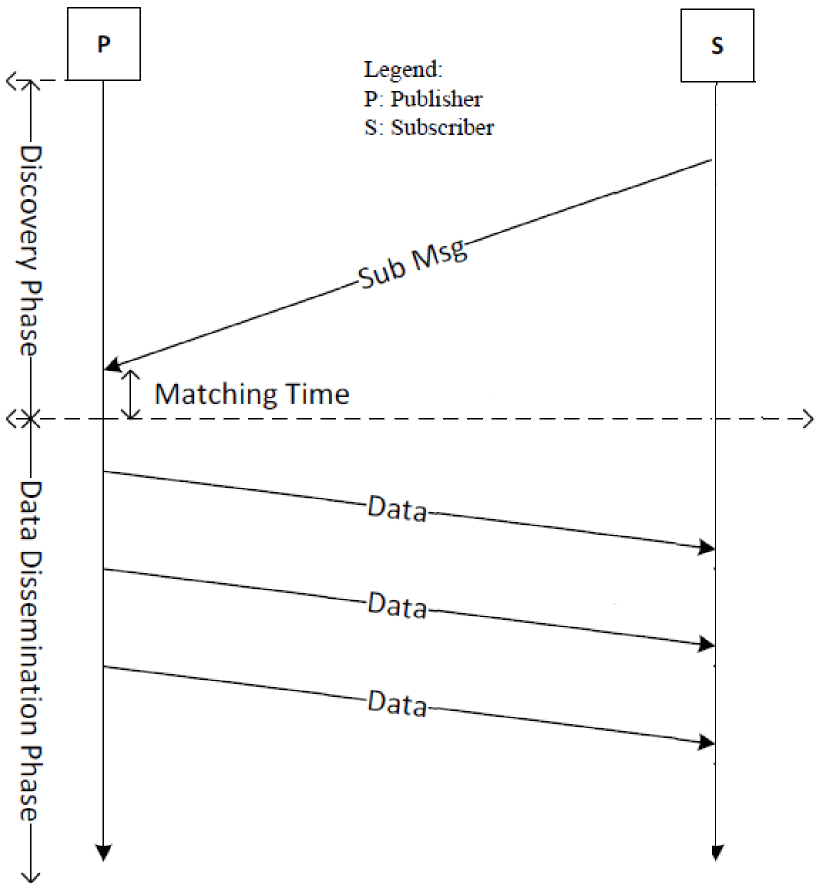


Figure 3.2: Discovery phase and Data Dissemination phase for Broker-less TinyDDS [23]

3.1.2 BLTDDS Architecture

Figure 3.3 depicts the architecture of BLTDDS middleware according to TinyDDS and the OMG DDS standard. It consists of four basic entities as follows: Application Programming Interfaces (APIs), the publisher, the subscriber, and the pub/sub service. The interface interact with every topic in the network using two main components: the Data Writer (DR), in the publisher side, and the Data Reader (DR), in the subscriber side. BLTDDS middleware intermediates between the application and the platform details, such as Sensor/Actuator complexity and TinyOS protocols.

Since the application only interacts with the system by the API and DDS interface, the application development becomes easier.

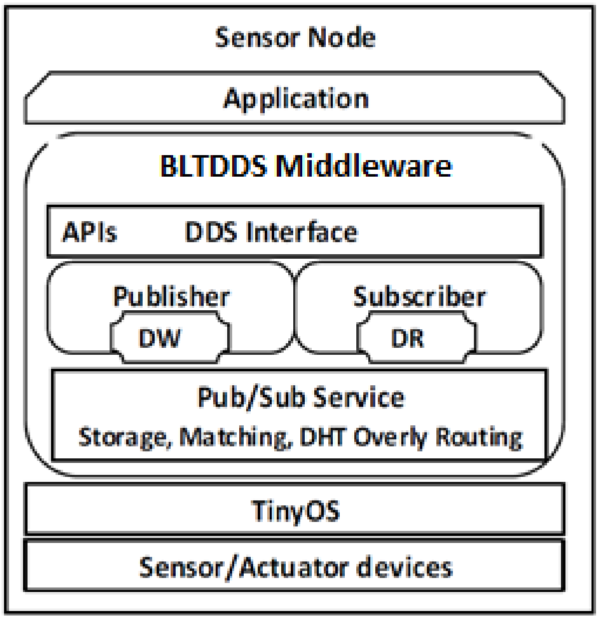


Figure 3.3:BLTDDS Architecture [23]

3.2 Simulation Tool

BLTDDS is implemented over TinyOS code. TinyOS is a framework which is designed for WSANs, and enable to build specific OS for each application. It’s a component-based model of programming using Network Embedded Systems C (nesC) language. On another hand, TinyOS SIMulator (TOSSIM) is an event-driven simulator, it's one of the most accurate and well-known tools to simulate the behavior of wireless sensor and actuator networks [30] [31].

TinyOS as a component-based operating system, it consists of many components that are wired using interfaces. For example in [32] an energy model was developed to overcome some of the TOSSIM energy calculation limitations, and it has two main components: Radio and MCU components.

Figure 3.6 illustrates the TOSSIM architecture, where it includes five parts: TinyOS compiling, simulation infrastructure, a discrete event queue, some TinyOS hardware support components, radio and ADC mechanisms, and communication services for external interaction [31].

TOSSIM generates discrete-event simulations based on TinyOS's structure and runs the same code used by sensor hardware. It translates the interruptions of the hardware into discrete events, then discrete event queue delivers them as interruptions to TinyOS applications [31].

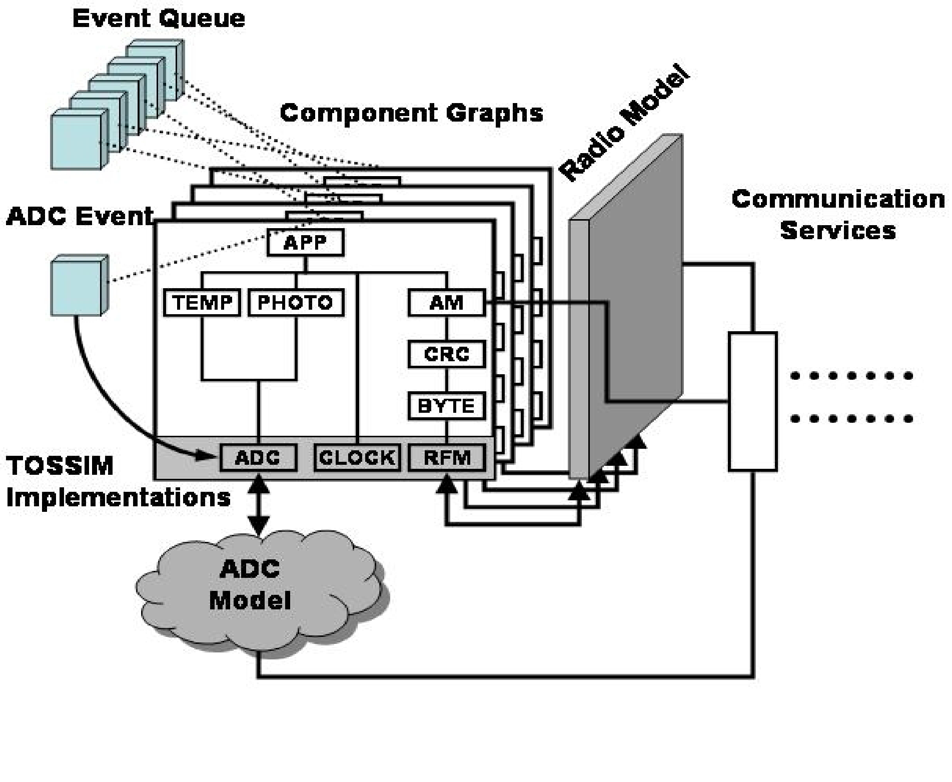


Figure 3.4: TOSSIM Architecture [31].

3.3 Performance Metrics

In this section, we discussed the performance evaluation Metrics, which are used to evaluate the behavior of the system application scenarios.

3.3.1 Packet Delivery Ratio

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. The larger the packets sent to the network, the larger the congestion, and buffer overflow occurs. If PDR is less than one, this means there is packet dropping in the system.

3.3.2 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. This delay includes transmission delay and queuing delay. It is expected that when the traffic load goes high then the queuing delay also goes high, as a result, the end-to-end delay will be increased. The delay is calculated for all successfully received messages by all subscribers and then the average is taken.

3.3.3 Energy Consumption

The power source of the sensors to work is batteries, for this reason, the power consumption is the critical issue in WSN, so, this type of networks require that the communication and all processes inside the systems work within minimum power consumption in order to maximize the lifetime of the node. The energy consumption is calculated by taking the summation of energy consumption of all the network nodes in milli-Joule. The radio and MCU are the only components that will be considered in our evaluation.

CHAPTER 4

DDS REAL-TIME QUALITY OF SERVICE (QOS) POLICIES IMPLEMENTATION

The DDS specification offers real-time policies to guarantee quality-of-service (QoS) in the network. Since BLTDDS is an improved lightweight version of DDS middleware for WSAN platforms, BLTDDS in the current form lacks the implementation support for DDS real Time QoS policies.

This chapter provides a detailed description for the implementation of DDS QoSs, (1) Time Based Filter, and (2) Deadline QoSs over BLTDDS. The new version of BLTDDS is called real-time BLTDDS (RT-BLTDDS).

4.1Time Based Filter QoS Implementation

In this section, a description of Time Based Filter (TBF) QoS is introduced. We describe in detail its implementation over pub/sub architecture, main components, and algorithms.

Time Based Filter (TBF) is a Quality of service policy which is not implemented in BLTDDS yet. According to OMG DDS, this QoS policy can be used by the Data Reader (DR) of each subscriber to reduce the amount of receiving data samples. This QoS is very useful, especially when the publisher Data Writer (DW) may send data samples at a rate faster than the Data Reader can receive due to resource limitation in sensor nodes. For example, in some applications the Data Reader is operating in human GUI application, in such cases the subscriber cannot receive data updates at a rate faster than the user can read the values and perceive the changes [18].

Data Writer can send data to different Data Readers with different capabilities, this means that Data Writer may send in a very fast rate, where the faster Data Reader can receive in the a proper way. The other Data Readers with slower receiving rates can still receive the updates with their receiving rates. For example, some data reader can read data every 0.1 seconds, and other ones may read data every 1 second, then the Data Writer should send each 0.1 second.

Using Time Based Filter QoS, different Data Readers can set their own Time based filter with the value that fit their requirements without affecting the sending rate of the Data Writer or affecting the receiving rates of other Data Readers. TBF can be applied for different instances separately, where the Data Reader does not want to receive more than one update sample from each instance per time separation.

In addition, TBF QoS policy allows for resource usage optimization (CPU, memory, network traffic and network capacity), where only the required amount of updated samples is delivered to each Data Reader. As a result, it can protect heterogeneous network application, where some nodes can generate data much faster than others can receive. Consequently, in the case of multiple Data Readers, the one with lowest separation time determines the Data Writer's publish rate. The minimum separation time the TBF provide, is the key rule for the application to work smoothly and to optimize the resources.

TBF looks like a switch, where Toff = minimum separation time and Ton = sending time. Figure 4.1 shows the switching concept of TBF. In case of two or more subscribers, each subscriber can request a distinct TBF QoS. When there are more than one instance updating one sample, TBF minimum separation time is applied per instance. In this case, the subscribing application will receive one sample from each instance per minimum separation time.

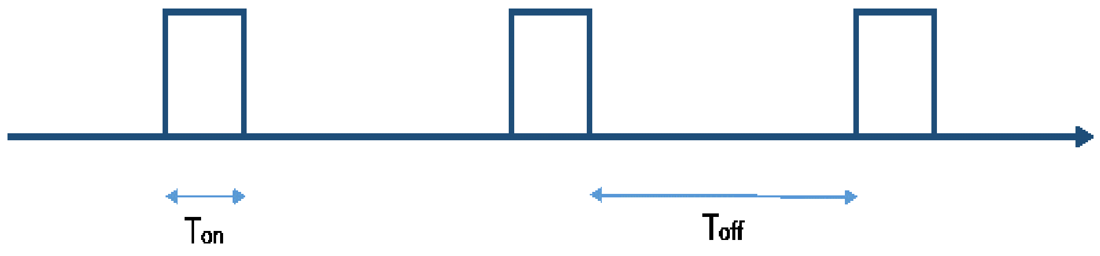


Figure 4.1: switching concept

Algorithm 1 shows how the TBF QoS is designed and implemented in the Data Writer component of the publisher and the requested TBF minimum separation time from Data Reader. Figure 4.2 shows the flowchart that describes the data sending and receiving behavior between Data Writer and Data Reader after TBF QoS is applied.

Algorithm 1: TBF QoS

```
Input variables : TBF, data_rate, Sim_time
Start Boot: Data Reader request for TBF QoS during subscription
phase.
Set TBF QoS in Data Writer.
initiate  App_Timer
While ( App_Timer<=  Sim_time) do:
            IF(is TBF set == true) Then:
                 initiate Timer;
                 While (Timer>0) do:
                         IF (Timer% TBF==0)  Then:
                               SendData( data_rate);
                               Reset Timer;
Timer++;
            Else:
             SendData ( data_rate);

            App_Timer++;
```

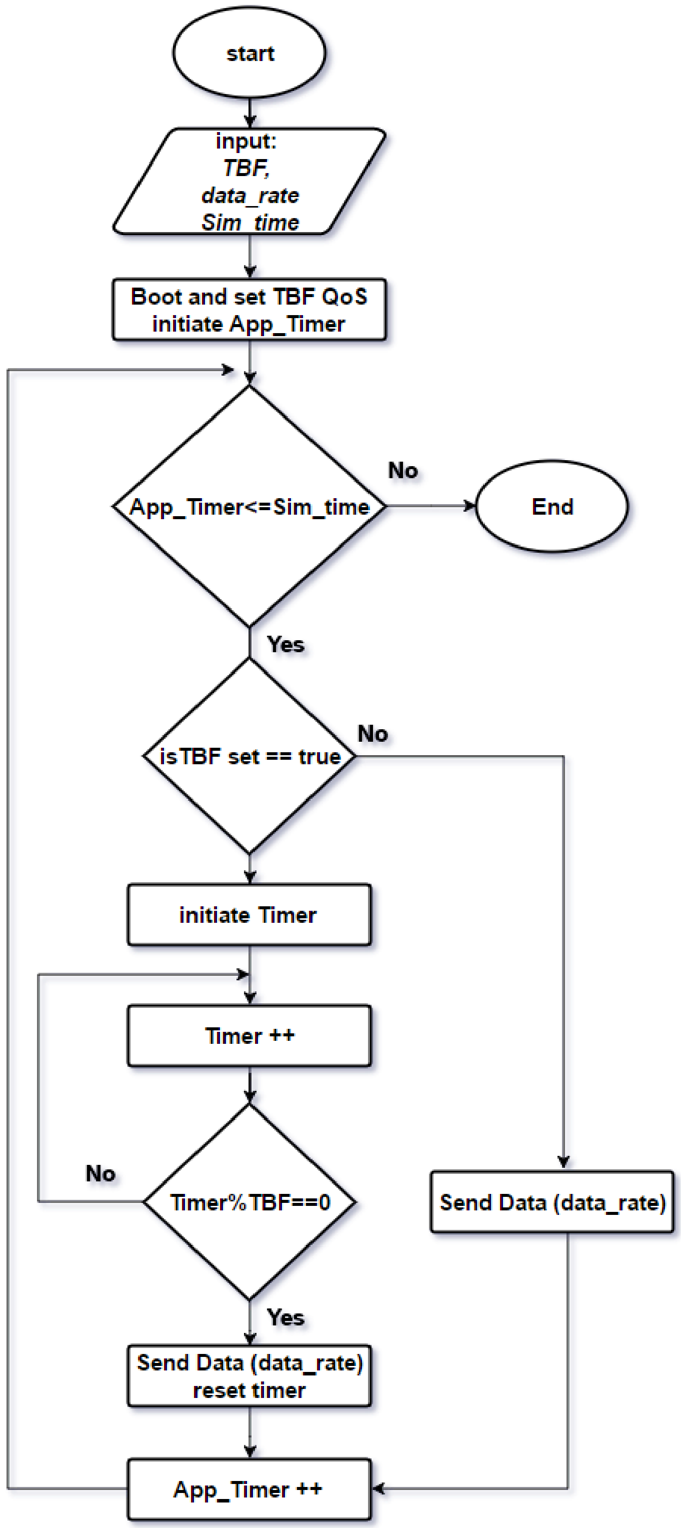


Figure 4.2: TBF QoS implementation flowchart.

4.2 Deadline QoS Implementation

In this section, a description of Deadline QoS is introduced. We describe in details its implementation over pub/sub architecture, main components, and algorithms.

According to DDS standard specifications, if this QoS policy concerns the publisher side, it is the connection contract that the application should meet to establish the connection. On the other hand, if this QoS policy concerns the subscriber side, it represents the minimum allowed time the publisher is expected to send the data values within [18].

Since that this QoS policy values determine if the connection will happen or not, the compatibility match on both sides should be checked upon this relation:

$$\textit{Offered deadline (DW)} <= \textit{requested deadline (DR)} \qquad (1)$$

Where: DW: Data Writer, DR: Data Reader

If this relation is not satisfied, the communication will not occur. Assuming that the publisher and subscriber have compatible settings, the fulfillment is monitored by proper component (listener) to inform the application for any violations [18].

However, in some important cases that we should be aware of, when Deadline, publishing rate, and the Time Based Filter minimum separation time are aligned, where missed deadlines accidents are expected to happen. Then TBF minimum separation time values should be close to the publishing rate, to avoid filtering more updated samples than the application require, then to send the critical data within this time.

In contrast, to avoid deadline missing, the TBF minimum separation time values shouldn't be too close to the requested Deadline. Otherwise, Deadline missing expected to happen. These scenarios demonstrate the consistent phenomena between the values of TBF minimum separation time and requested/offered Deadline QoS's policies. This phenomenon can be expressed with the following relation:

$$DR\ Deadline >= DR\ TBF_{minimum\ separation} + DW\ Deadline \qquad (2)$$

Where: TBF: Time Based Filter, DW: Data Writer, DR: Data Reader

The default value for Deadline QoS policy is infinity; however, if it is set to a specific value which is not infinity, it directly defines the maximum Inter-Arrival Time between data samples on the subscriber side. For example, this offered Deadline QoS is very important for cases of real-time monitoring applications, i.e. rocket tracking. Where the publishers should offer less deadline, at which the data should be available within to be sent to the subscriber or base station. Also requested Deadline should be set to a value that no critical data may lose.

Suppose that we have one subscriber and one publisher updates the topic instance, e.g. temperature. Assume that the publishing rate is one sample per second and the Deadline QoS is set to be one second, which means the Data Writer of the publisher must publish one sample of the instance per one second. Then the Data Reader of the subscriber will receive one updated samples per one second. If the Data Reader Deadline QoS set to be 4 seconds with neglecting the delay of transmission, then we would have 2 cases where the Data Reader TBF QoS is set to be 5 seconds in the first case, and 3 seconds in the second case.

In the first case, the DR TBF is equal to 5, if we are applying the relation number (2) then summation of TBF and DW Deadline is equal 6 (1+5) which is not less than or equal to 4. In this case, the consistency does not happen and the connection between the publisher and the subscriber will not be established.

In the second case, the TBF is equal to 4, then after applying the relation the result is 4 (1+3) which is equal to DR Deadline. In this case, the connection will happen and the data will be sent after TBF minimum separation time.

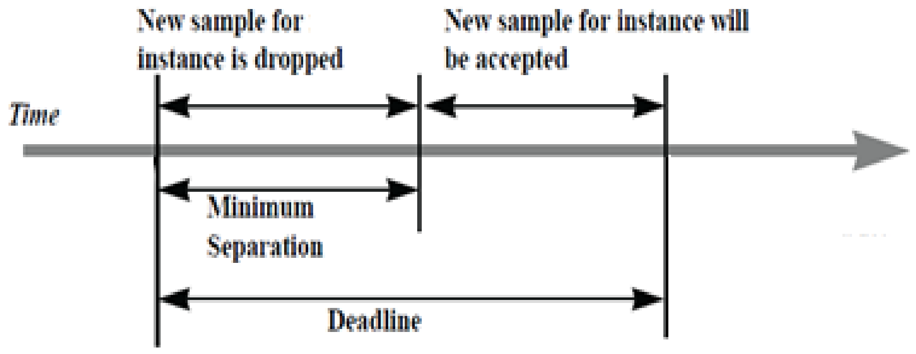

New sample for instance is dropped
New sample for instance will be accepted
Time
Minimum Separation
Deadline

Figure 4.3 QoS's policies in Data Reader side

Algorithm 2 shows how the Deadline QoS is designed and implemented in the Data Writer component in the publisher side. For TBF QoS, algorithm 3 shows how the Deadline and TBF QoSs are designed and implemented together. Figure 4.4 and Figure 4.5 show the flowchart that describes the data behavior after Deadline QoS is applied. However, in Figure 4.5 the consistency issue between TBF and Deadline QoS policies are considered.

Algorithm 2:Deadline QoS

```
Input variables : deadline, Req_deadline, data_rate, Sim_time
Start Boot: Data Reader request for deadline QoS during
subscription phase.
Set Req_deadline QoS in Data Writer.
Set deadline (i)QoS in Data Writer for each publisher.
While ( App_Timer <=  Sim_time) do:
        For ( i=1, i <=Publishers, i++)do:
IF (deadline (i)<= Req_deadline)  Then:
                                        SendData (data_rate);
App_Timer++;
```

Algorithm 3: Deadline and TBF QoS's implementation

```
Input variables : deadline, Req_deadline,TBF, data_rate, Sim_time
Start Boot: Data Reader request for TBF and Deadline QoS during subscription phase.
Set TBF QoS in Data Writer.
Set Req_deadline QoS in Data Writer.
Set deadline (i) QoS in Data Writer for each publisher.
initiate App_Timer
While ( App_Timer <= Sim_time) do:
                IF(is TBF set == true) Then:
        For ( i=1, i <= Publishers, i++)do:
                IF (deadline (i)+ TBF<= Req_deadline) Then:
                      initiate Timer;
                      While (Timer>0) do:
                                IF (Timer% TBF==0) Then:
                                        SendData ( data_rate);
                                        Reset Timer;
Timer++;
                Else:
        For ( i=1, i <= Publishers, i++)do:
                IF (deadline (i) <= Req_deadline) Then:
                SendData ( data_rate);


App_Timer++;
```

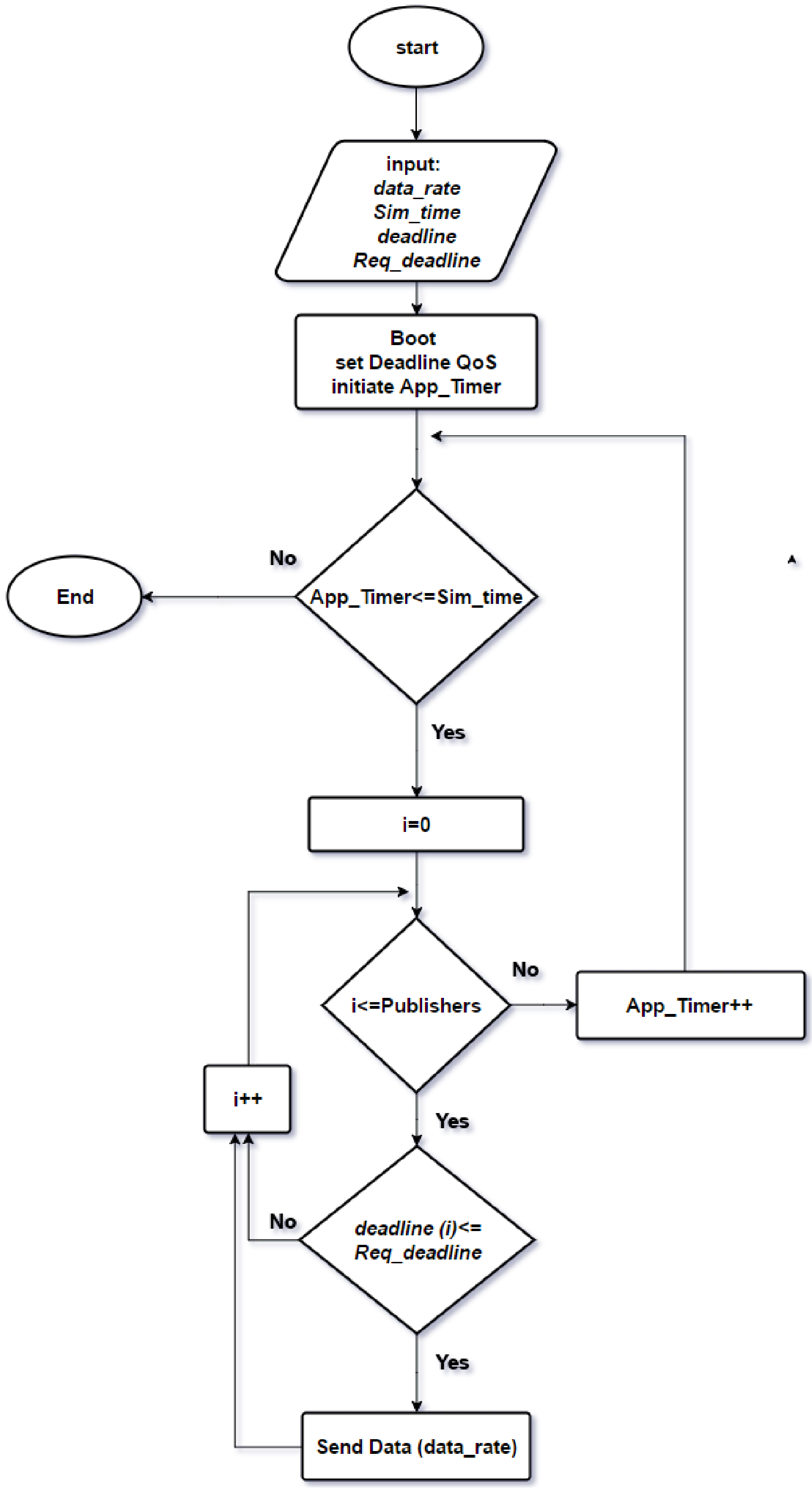


Figure 4.4: Deadline QoS implementation flowchart.

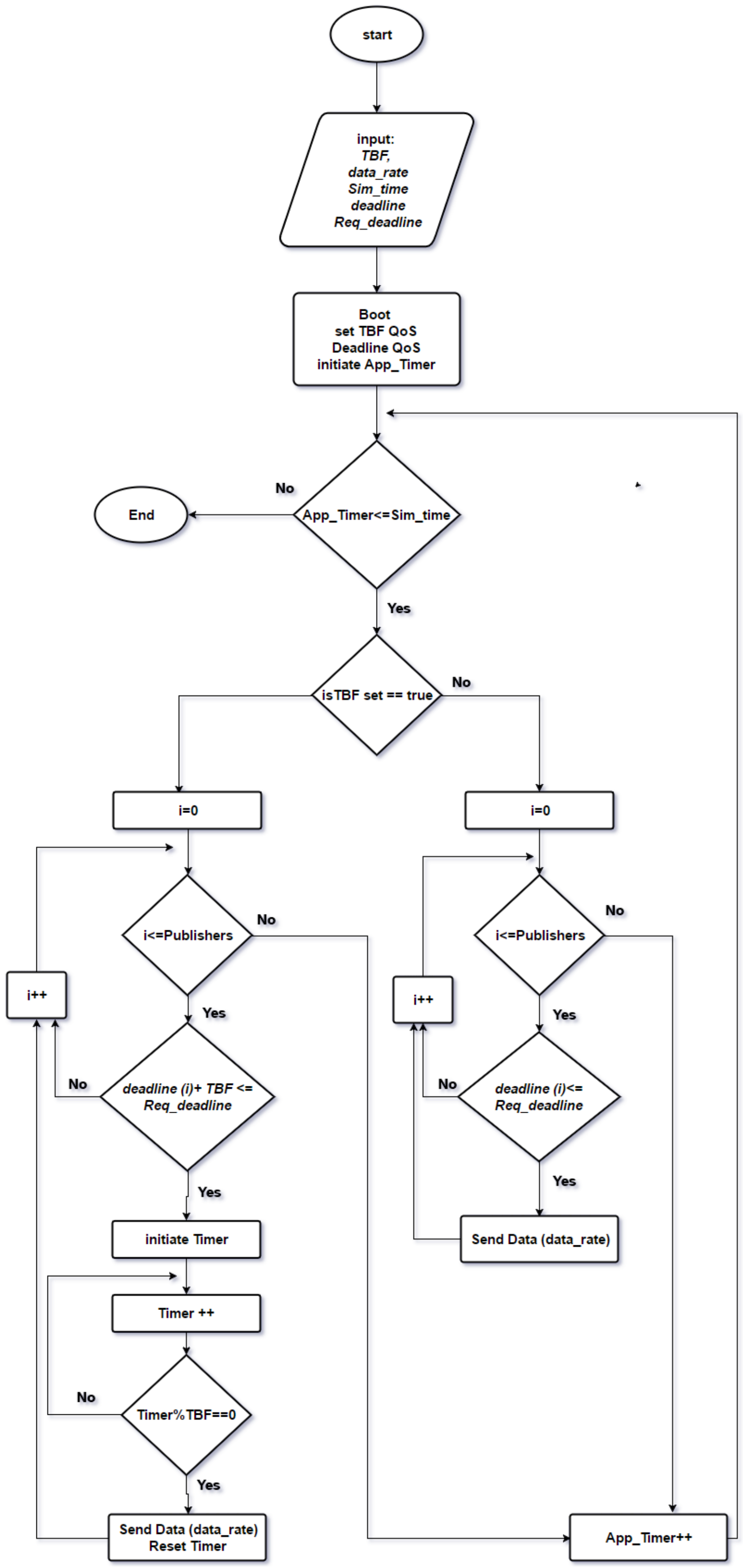


Figure 4.5: Deadline and TBF QoS's implementation flowchart.

CHAPTER 5

PERFORMANCE EVALUATION

In order to have an accurate comparison between our work and the previous work, we generated base line results (the default BLTDDS before adding the QoSs) that is used to evaluate our proposed work (BLDDS after adding QoSs).

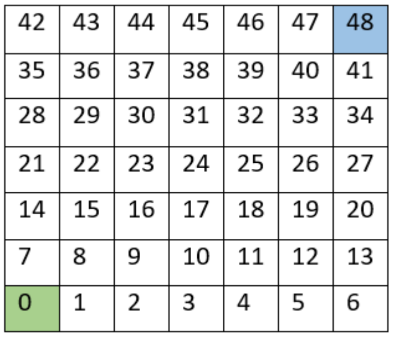
In this chapter, we evaluate our work performance, and discuss the simulation setup, and the results and analysis of our proposed work.

5.1 Simulation Setup and parameters

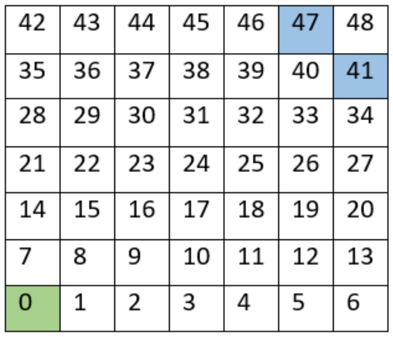
According to previous work, the grid topology for WSANs was used extensively to simulate the behavior and the distribution for sensor and actuator nodes in practice. In this work, we use the same scenarios of a grid topologies with different distributions for the publishers. However, we use just one base station (subscriber) since it's the usual and dominant situation for most of all WSANs applications.

5.1.1 Application Scenario

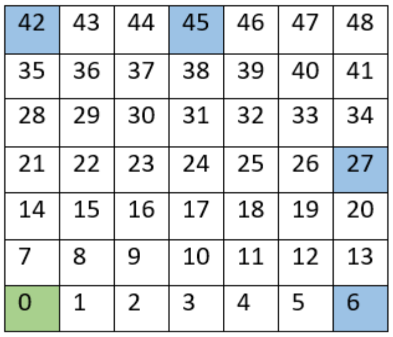
In this case, we considered one of the experimental scenarios done in this work, which is a grid topology of 49 nodes distributed uniformly in an area of 100 ×100 m^2. In this network, we set one node to be the subscriber (the Base Station), then the number of publishers is changed from 1, 2, 4, 8, 16, 32, 40, and 48 (full load) to evaluate the performance of the network and test the scalability.



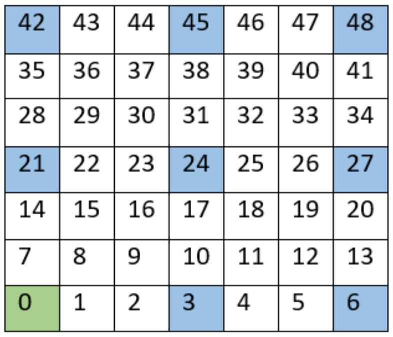
A : 1 subscriber, 1 publisher



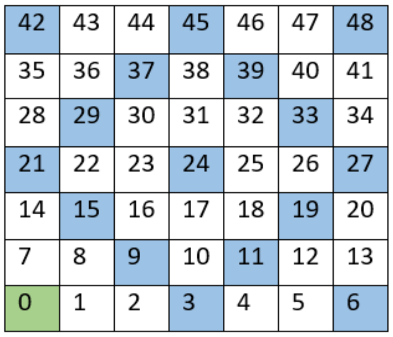
B : 1 subscriber, 2 publishers



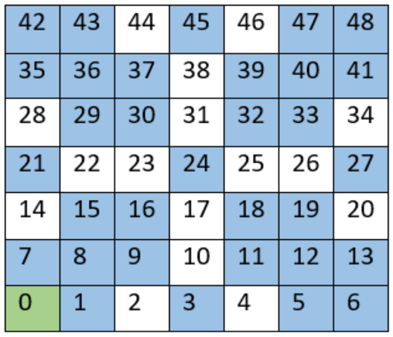
C : 1 subscriber, 4 publishers



D : 1 subscriber, 8 publishers



E : 1 subscriber, 16 publishers



F : 1 subscriber, 32 publishers

Figure 5.1: publisher nodes distribution over the area represented by blue color cells, and the Base-station subscriber is represented by green color cell. The cells with the white color represent the relay nodes used for packets forwarding purpose.

We use one topic to find out the effect of increasing the publisher nodes on the network performance. Figure 5.1 shows the different distribution forms of the subscriber and the publishers in the network. For example, in the first scenario, one node at the top right corner with id 48 is the sender/publisher, and the remaining nodes are relay nodes. Thereby, the maximum number of hops is nearly 10 hops, sometimes due to network congestions/failures the routing protocol selects longer paths. The remaining nodes are relay nodes used for packets forwarding purpose. Table 5.1 describes the common parameter used to simulate this scenario before and after implementing the QoSs.

Table5.1: simulation setup for the tested application scenarios.

Parameter	**Value**
Topology	Squared grid
Area	100 X 100 Meter^2
Number of Nodes	49
Simulation time	500 seconds
Radio model	Chipcon CC2420
Mote platform	micaZ
Data rates	1, 2, and 4 packets /s
Number of publishers	1,2, 4, 8, 16, 32, 40, and 48
Sample size	I packet
Packet size	20 bytes
Maximum hops	10
Runs per results' data point	10

The application that is used in this scenario acts as a collector for some reading values from the surrounding environment such as temperature, pressure, or humidity. This application is used in four scenarios. The first scenario uses the default BLTDDS without adding any QoSs. In the second scenario, we added only the Time Based Filter QoS Policy to BLTDDS, whereas in the third scenario we added only the Deadline QoS policy to BLTDDS. Bothe QoS policies were added into BLTDDS in the fourth scenario. The results from the first scenario will be used as Base line, and all other scenarios' results will be compared with it. Figure 5.2 shows the flow chart for the application behavior.

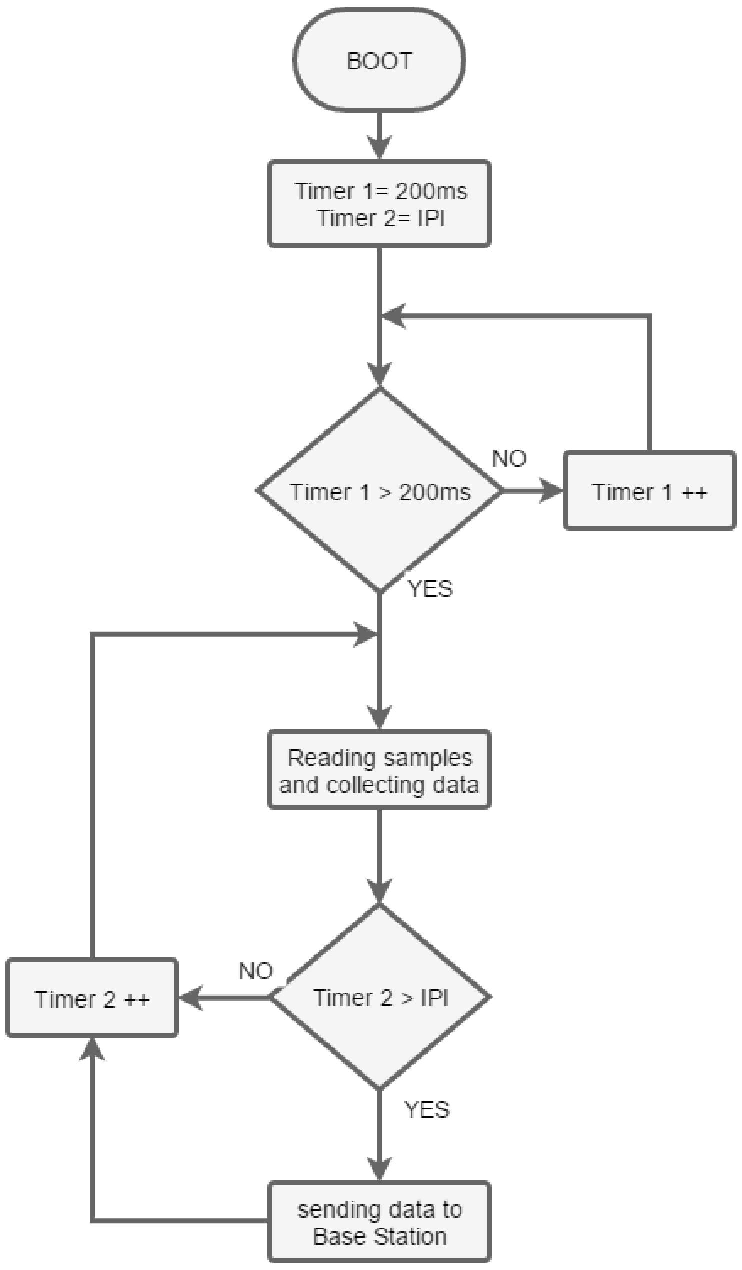


Figure 5.2: The basic application algorithm

5.2 Results and analysis

5.2.1 Base Line

The performance evaluation of BLTDDS is conducted using the three metrics, which are Packet Delivery Ratio (PDR), Latency, Processing and Radio Energy Consumption. During this evaluation the number of publishers is changed (i.e. 1, 2, 4, 8, 16, 32, 40 and 48 publishers (full load)) and the data rate varies (i.e. 1P/s, 2P/s, and 4P/s).

Packet Delivery Ratio (PDR) Percentage

Figure 5.3 shows the effect of the network load on the PDR when changing the Data Rate. We notice that the PDR is decreasing as the number of publishers is increasing. That is because in case of one publisher there were no other publishers that may use the same path; in the other case, Figure 5.1(b), two publishers are close to each other, and may publish in different or in the same time. Since they are close to each other, they may use the same path for packet forwarding toward the subscriber and this may increase the packet dropping in the network which in turn decreases the PDR. If the publishers start publishing in different times this will increase the PDR due to less packet collisions and dropping in the network.

On the other hand, in case of four publishers, where they are away from each other and the probability to use different paths is high so that the number of dropped packets will decrease and PDR is increased. In case of 8, and 16 publishers, the network becomes denser and the dropping increases which results in decreasing the PDR. In case of 32 publishers, the PDR increases slightly since the number of received packets increases, however it decreases after that when the network is fully loaded. In general, increasing the number of publishers will decrease the PDR percentage. The results also show that increasing the data rate results in decreasing the PDR percentage; that is because increasing the data rate will increase the network traffic, which increases the probability for high packet dropping, also the difference between different data rate curves increases when the number of publishers increases.

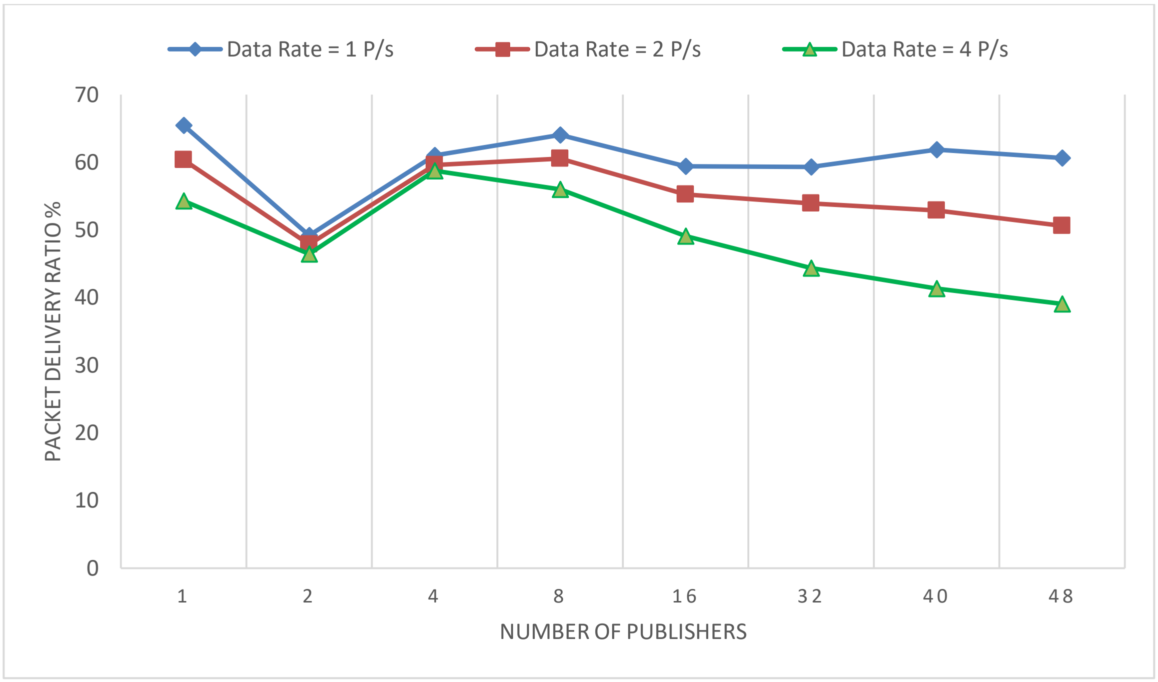


Figure 5.3: PDR behavior in different data rate (1p/s, 2p/s, and 4p/s) while changing the number of publishers.

Latency:

Figure 5.4 shows that the delay is almost the same for increasing the data rate and a slight difference can be noticed in case of more than 16 publishers. As shown in the figure, the delay decreases when the number of publishers increases, and this happens because in case of one and two publishers the distance is maximum to the subscriber, which increases the number of hops needed to forward the data and a lot of processing delay is added to the total delay.

However, increasing the number of publishers from 8 to 32 will decrease the delay. That is because increasing the number of publishers will minimize the distance between the new publishers and the subscriber (base station). In average, the distance from the publishers to the base station will be decreased as we increase the number of publishers. In this case, the path is shorter and number of hops are decreased which decreases the total delay for those close publishers to the subscriber. In addition, it decreases the average delay as we see in the figure since more packets are received from the publishers, which are close to the subscriber location in comparison to those that are farther.

For more than 32 publishers, we see that the average latency is increasing because the network load becomes full; the reason is that there is a lot of collisions and wireless interference that affects the transmissions. Moreover, high number of packets will share the same path which add more overhead to the nodes to forward them and more queuing and processing delay.

Figure 5.5 shows the changing of packets latencies for 16 publishers related to the time. From this figure we conclude that the Mean = 25.77458 ms, Standard Deviation= 11.1938 ms, max= 62 ms, and min = 3 ms. The statistical analysis shows that 99.04% of delivered packets arrived in less than

50 ms. 95.66% of delivered packets arrived in less than 45 ms. 89% of delivered packets arrived in less than 40 ms, and 80% of delivered packets arrived in less than 35 ms.

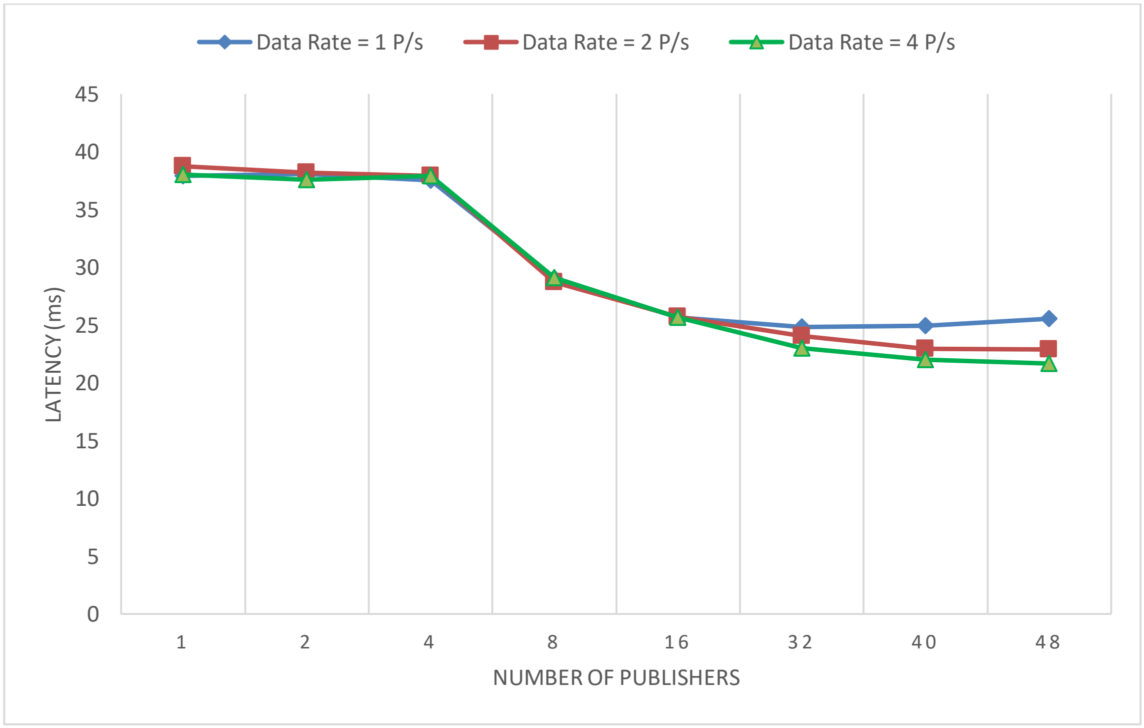


Figure 5.4: latency behavior with different data rates (1p/s, 2p/s, and 4p/s) and different number of publishers.

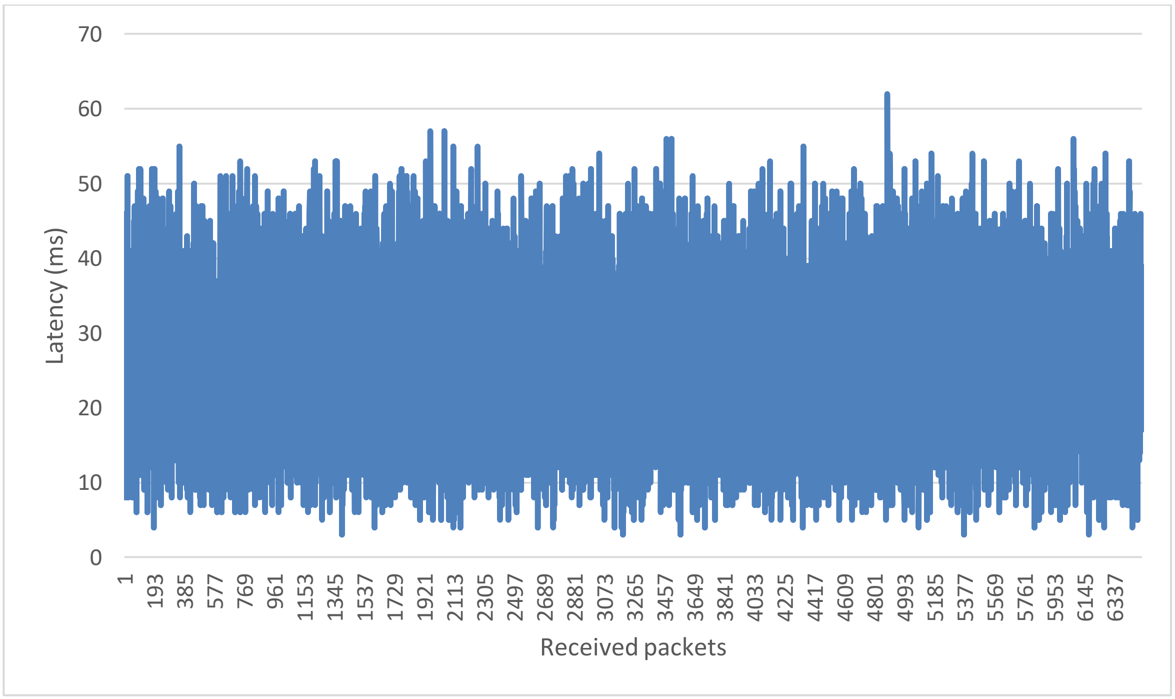


Figure 5.5: delay behavior for the received packets related to the time in case of 16 publishers.

Total Processing and Radio Energy Consumption:

Figure 5.6 shows the total energy consumed by all nodes in the network; this energy represents the system CPU usage of all the network nodes. Since we use the same node platform, which is micaZ, for all network nodes, this result shows a good scope to compare the results in case of increasing the number of publishers. We noticed that the total processing energy consumed is in a scale of micro joules.

When the number of publishers were increased the total consumed energy was increased, also increasing the data rate increases the overall processing energy. Where the increasing is about 0.19% from 1 publisher to 48 publishers and the data rate is 1 p/s. Also, 0.21% increasing in total processing energy when we change the number of publishers from 1 to 48 in data rate = 2 p/s. Moreover, 0.22% increasing in energy usage by CPU of all nodes when we change the number of publishers from 1 to 48 in data rate = 4 p/s.

In addition, the increasing in total nodes CPU energy consumption when we change the data rate from 1 p/s to 2 p/s is about 0.0065 % in case of 1 publisher and 0.03 % in case of 48 publishers.

Figure 5.7 shows the total energy consumed in radio by all nodes in the network. From the results, we see significant change in the energy consumption of radio transmission in both cases of increasing the number of publishers or increasing the data rate. The scale of change is in mille-joule.

When we increase the number of publishers from 1 to 48, the increasing in the total radio energy consumption is 7645.014 mj and 12899.679 mj, in case of 1 p/s, 2 p/s, respectively. We conclude that the most energy consumption is by radio transmission, and the energy consumed by CPU is negligible in comparison.

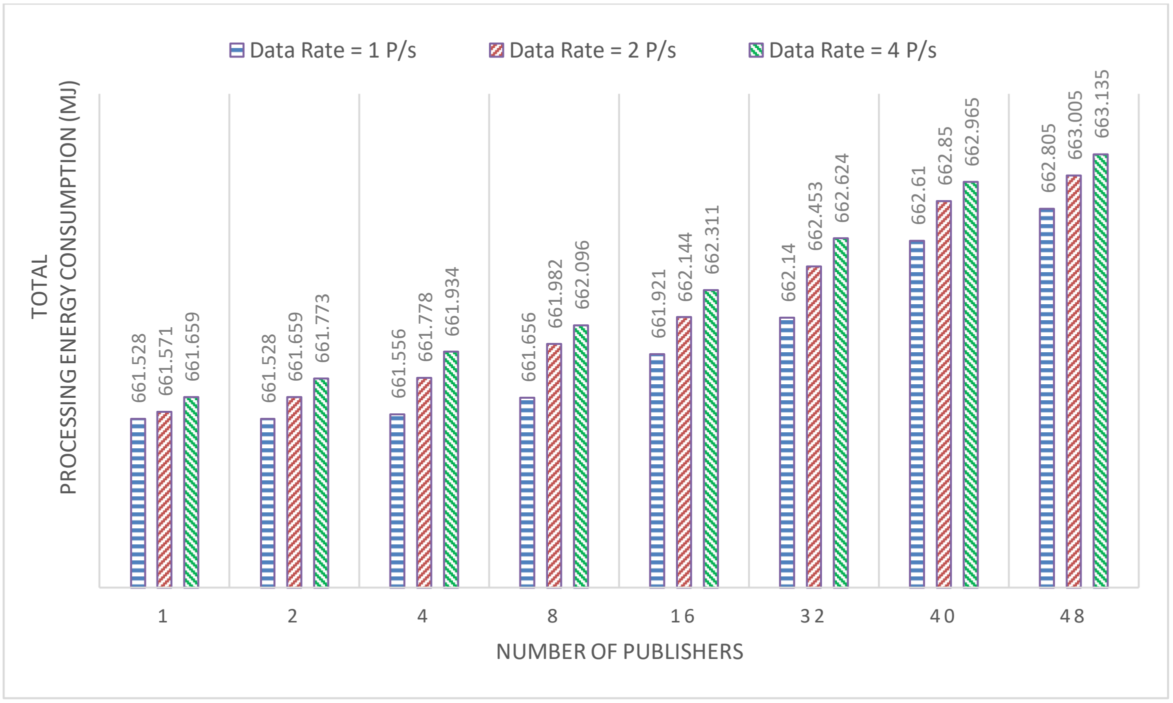


Figure 5.6: The Total Processing Energy consumption behavior in different data rates (1p/s, 2p/s, and 4p/s), and with different number of publishers.

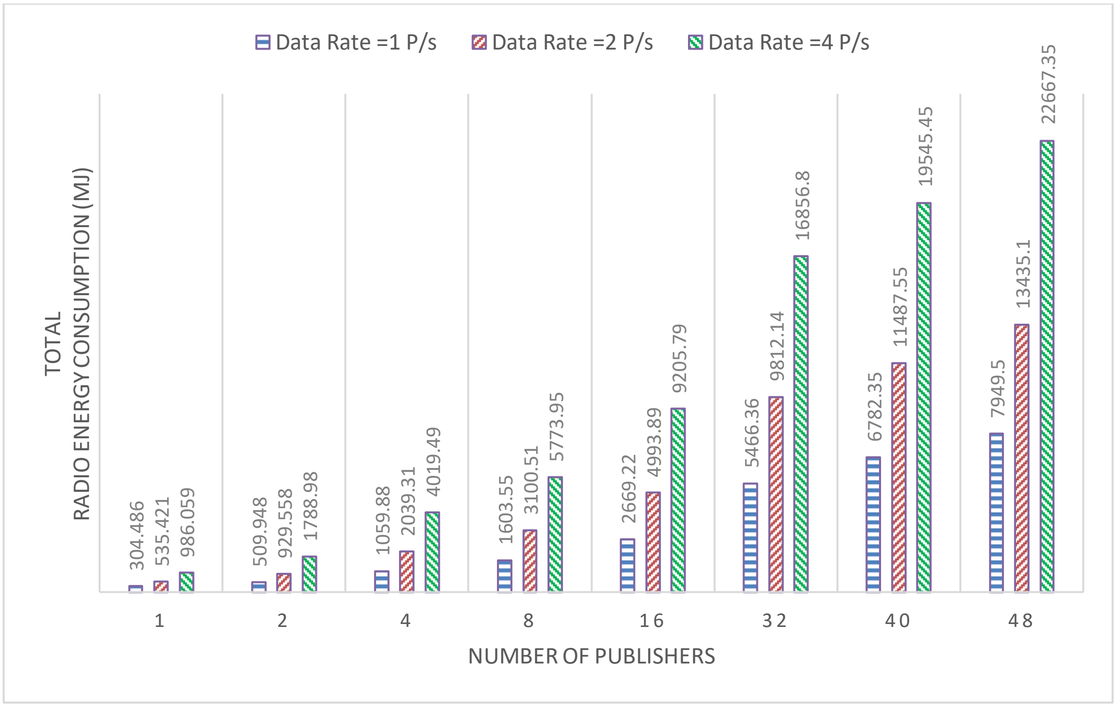


Figure 5.7: The Total Radio Energy consumption behavior in different data rates (1p/s, 2p/s, and 4p/s), and with different number of publishers.

5.2.2 Time Based Filter QoS Policy results

In this section, we figure out the effects of adding the Time Based Filter QoS to the system and to evaluate the performance afterwards. The results after simulating and testing is compared with the previous results of the Base Line (default BLTDDS). We use the same metrics for comparison and testing the behavior of the network as follows:

Packet Delivery Ratio (PDR) Percentage

From Figure 5.8 and 5.9, which show the network packet delivery ratio percentage when changing the Time Based Filter (TBF) minimum separation time from 2, 3, to 4 and the results compared with base line results in case of 1 p/s and 2 p/s data rates, and changing the number of publishers from 1 to full load.

From the figure, we notice that the effect of TBF in case of 1, and 2 publishers are the same in comparison with the base line, however it starts changing after 4 publishers. The results show that the effect of Time Based Filter QoS is improving the Packet delivery ratio and it is clear in case of more publishers that the PDR is increasing as much as the TBF minimum separation time values are increased. That is because when we increase the time between successive sent packets to suite the limited resources in the subscriber side, the publishers will not publish until TBF minimum separation set by the subscriber is applied. When the value of minimum separation time is increased, the available network bandwidth increases, because less packets will be delivered compared to the case where the filter is not applied. Decreasing the published data will affect directly the network behavior, where the performance of the network will be improved when less data will be published. Since TBF QoS decreases the network load over the network nodes in forwarding and delivering data, packet delivery ratio increases as a result. Also, we notice that in case of 32 publishers, when TBF minimum separation time is 2 seconds and data arte is 1 p/s, the system performance is improved by 8 %, and 15 % in case of 2 p/s data rate. When TBF is 3 seconds the system performance improved by 11.1 % in 1 p/s data rate, and 23% in 2 p/s data arte.

The improvement when the TBF minimum separation changes from 2 to 3 seconds is 3.1 % in case of 1 p/s and 8 % in case of 2 p/s data rate. However, from 3 to 4 seconds is 2.89% in 1 p/s data rate and 5.9 % in 2 p/s. The reason is that when TBF = 2 s applied, the decreasing in published data is more than when TBF=3 s. Thus increasing TBF minimum separation time will not improve the performance in the same rhythm but the improvement will be less. On the other hand, increasing TBF will decrease the publish data which is not preferred.

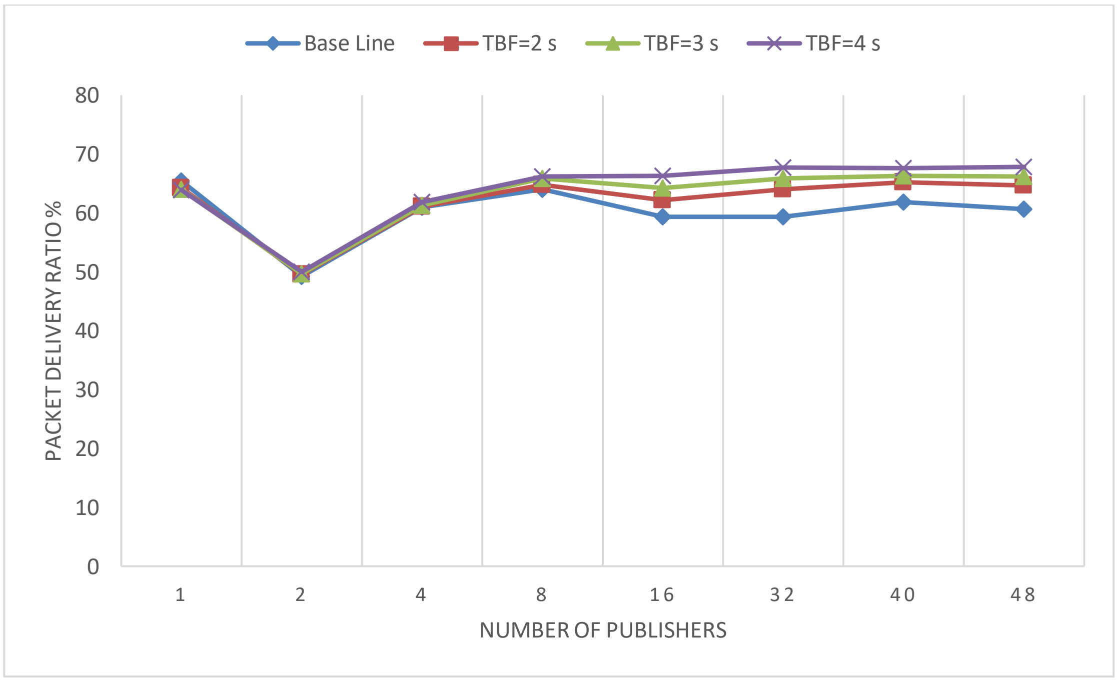


Figure 5.8: PDR behavior in different TBF minimum separation (2 s, 3 s, and 4 s)while changing the number of publishers in data rate of 1 p/s.

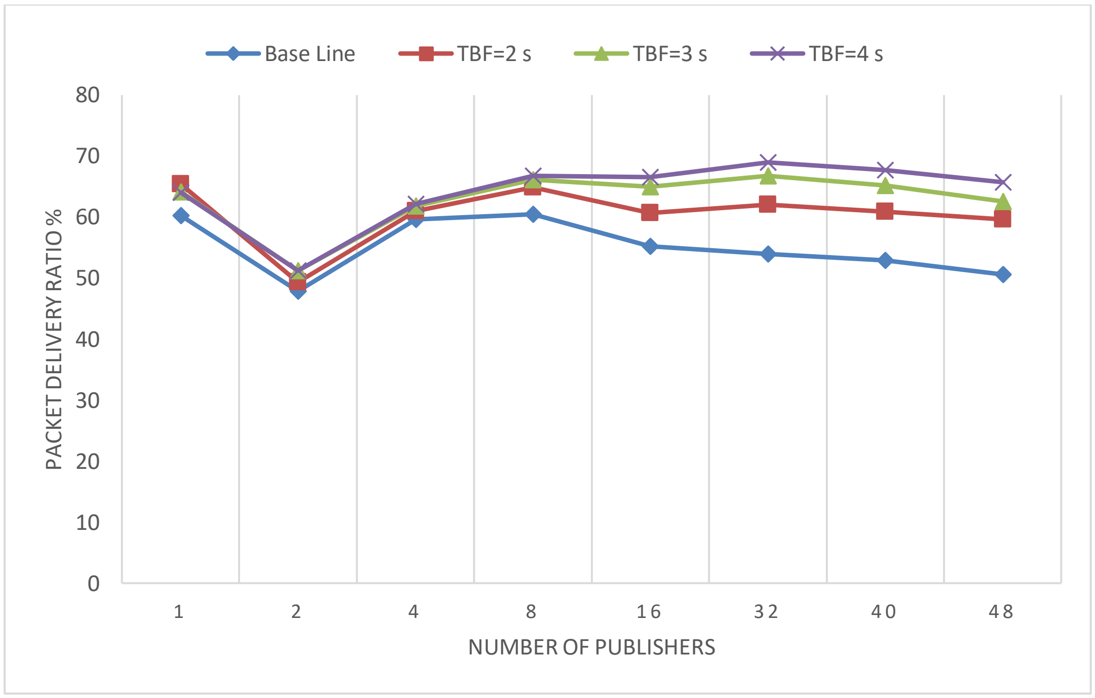


Figure 5.9: PDR behavior in different TBF minimum separation (2 s, 3 s, and 4 s) while changing the number of publishers in data rate of 2 p/s.

Latency:

Figures 5.10 and 5.11 show the latency result behavior after applying TBF QoS in different data rates (1, 2 p/s). We notice that the latency is decreasing while increasing the number of publishers. However, when the number of publishers is 32 and delay starts to increase since more publishers will increase the effect of wireless interference and packet collisions.

TBF QoS implementation is improving the latency behavior as the results illustrate in Figures 5.10 and 5.11, where increasing TBF minimum separation time will decrease the average latency in the network as we increase the number of publishers to 32 publishers. However, adding more publishers will dense the network much more and the average latency will increase, since more delay in queues and processing will be added to the network overall latency.

In case of 32 publishers, when TBF minimum separation time is 2 seconds, the latency improvement is about 3.6% in case of 1 p/s and 3.9% in case of 2 p/s. When TBF = 3 s, the improvement is about 5.6% in both data rate. When TBF=4 s the system latency is improved by 7%. This improvement is decreasing when increasing the TBF minimum separation time values. The effect of TBF QoS over the system delay will decrease when the number of publishers is increased more than 32. The system is scalable although the TBF QoS guarantee more system performance in less dense network.

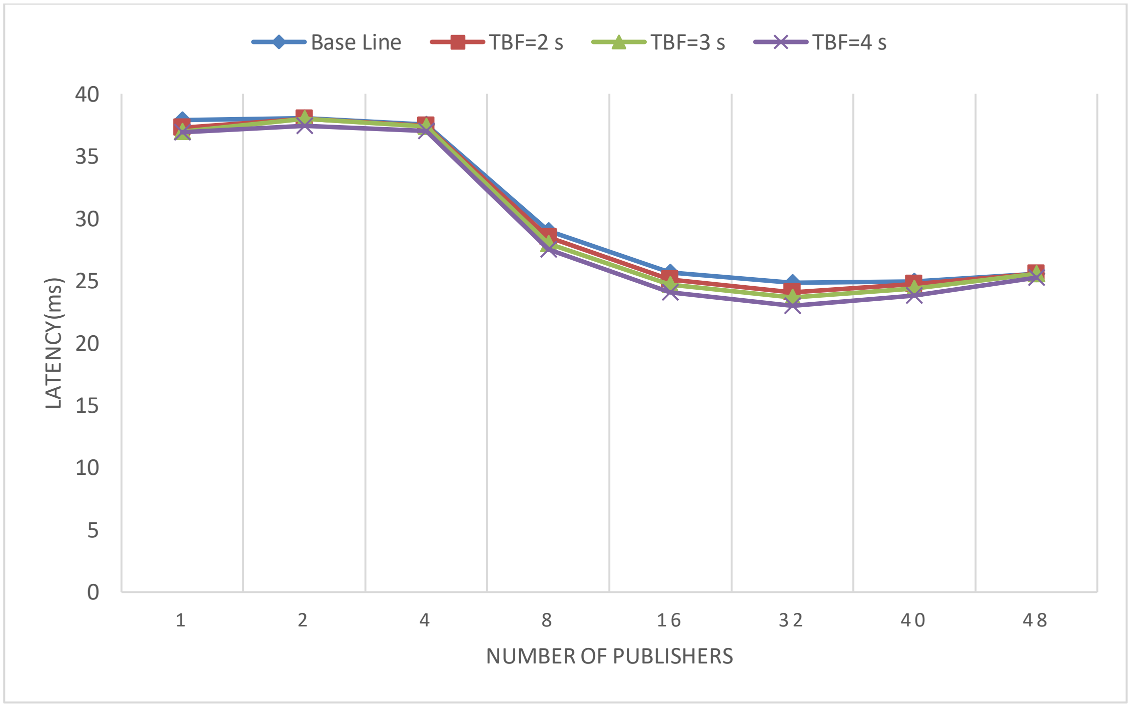


Figure 5.10: System latency behavior in different TBF minimum separation (2 s, 3 s, and 4 s) while changing the number of publishers in data rate of 1 p/s.

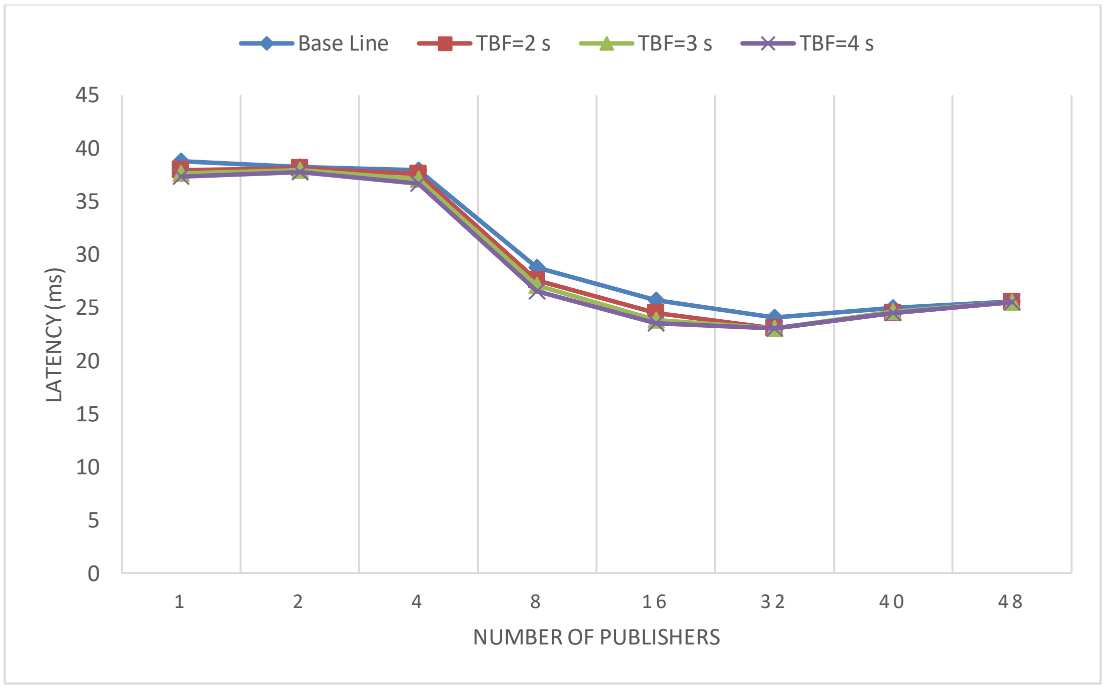


Figure 5.11: System latency behavior in different TBF minimum separation (2 s, 3 s, and 4 s) while changing the number of publishers in data rate of 2 p/s.

Total Processing and Radio Energy Consumption:

Figure 5.12 and 5.13 show the results after applying TBF QoS over energy consumption in the system and they compare it to the base line results for both energy type processing and radio in case of 1 p/s data rate. In addition, Figures 5.14 and 5.15 show the effect of applying TBF QoS in both types of energy when data rate is 2 p/s.

From Figures 5.12 and 5.14 we notice that the total processing energy consumed by all nodes in the network are decreased in case of 2, 3 and 4 seconds of TBF minimum separation and this return to the effect of decreasing the packet forwarding mechanism since the publishing from all publishers is decreased. Added to that, the improvement in the total consumed processing energy was significant when increasing the number of publishers. For example, when number of publishers are 48, when TBF = 2 seconds the total processing energy consumed by all nodes decreased by 0.11 % in 1 p/s and 0.1 % in 2 p/s. when TBF= 3 seconds the total processing energy consumed by all nodes decreased by 0.12 % in 1 p/s and 0.115 % in 2 p/s.

In Figures 5.13 and 5.15 the results clarify that applying TBF QoS decreases the total radio energy consumption significantly. The reason is that applying TBF QoS decreases the amount of traffic in the network and this will lead to less transmission in wireless medium that affect directly the radio energy for each node. The improvement is proportional to the number of publishers as we see when number of publishers are 1 the total energy consumed by radio is decreased by 39 % when TBF = 2 seconds and data rate =1 p/s. However when number of publishers are 48 the total energy consumed by all nodes in radio is decreased by 46 % for the same data rate. For the same TBF when data rate is 2 p/s, the total energy consumed by all node radio is decreased by 43 % for 1 publishers and 41.3 % for 48 publishers.

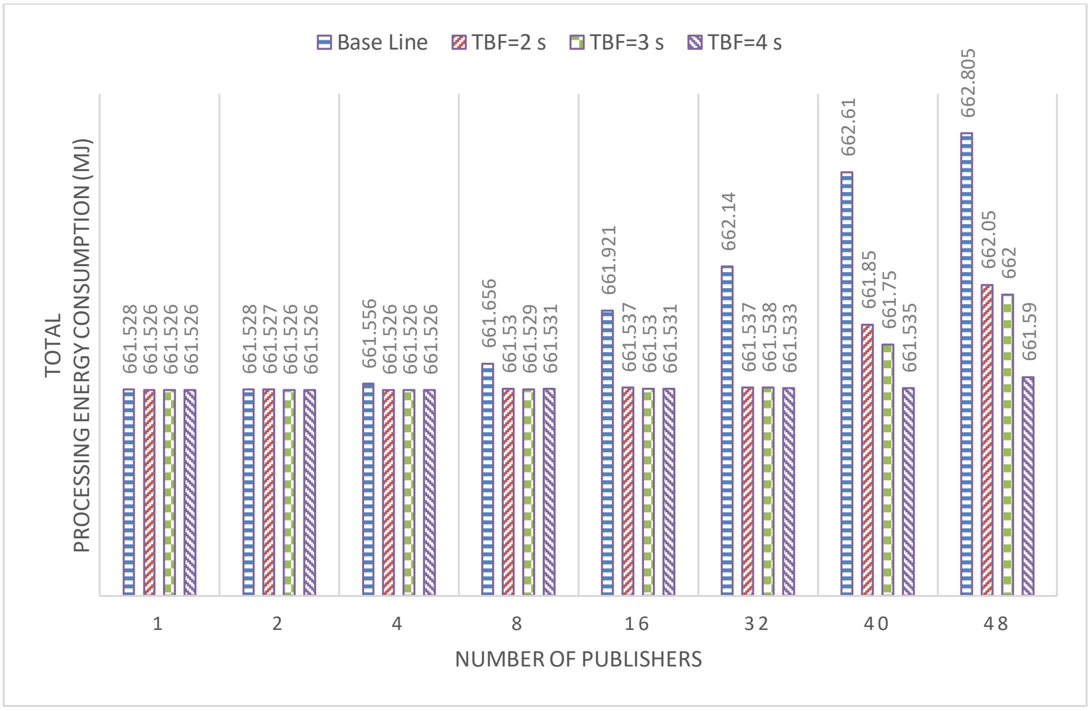


Figure 5.12: Total consumed processing energy for different TBF minimum separation (2 s, 3 s, and 4 s) while changing the number of publishers in data rate of 1 p/s

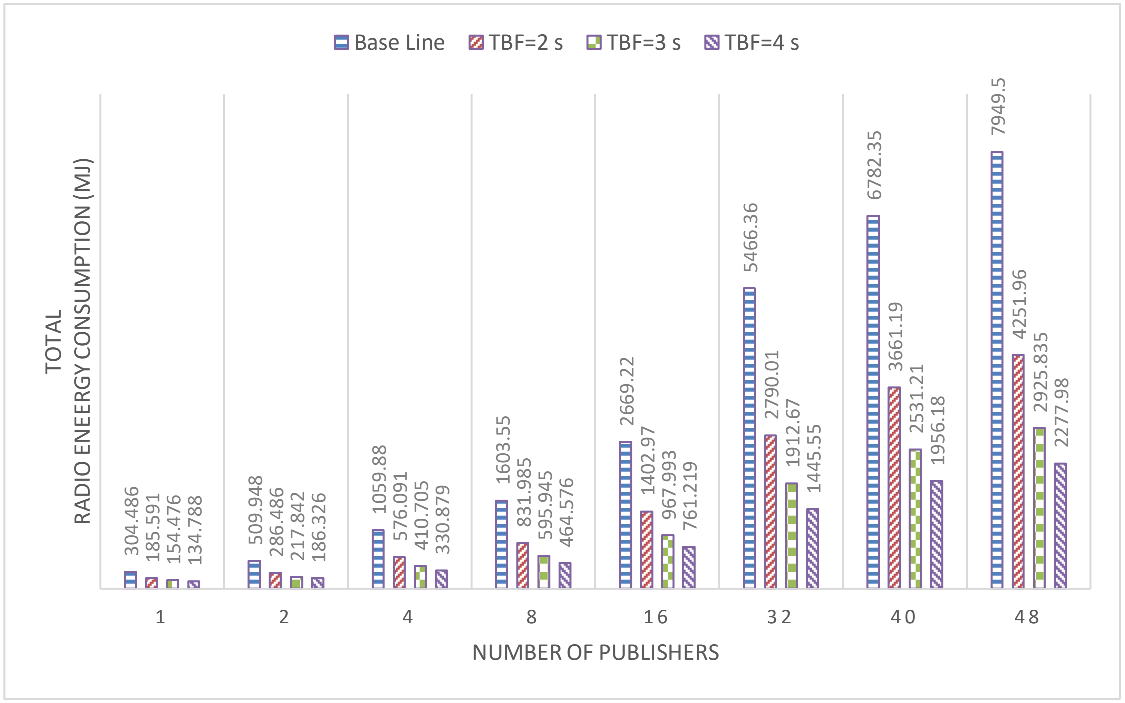


Figure 5.13: Total consumed radio energy for different TBF minimum separation (2 s, 3 s, and 4 s) while changing the number of publishers in data rate of 1 p/s.

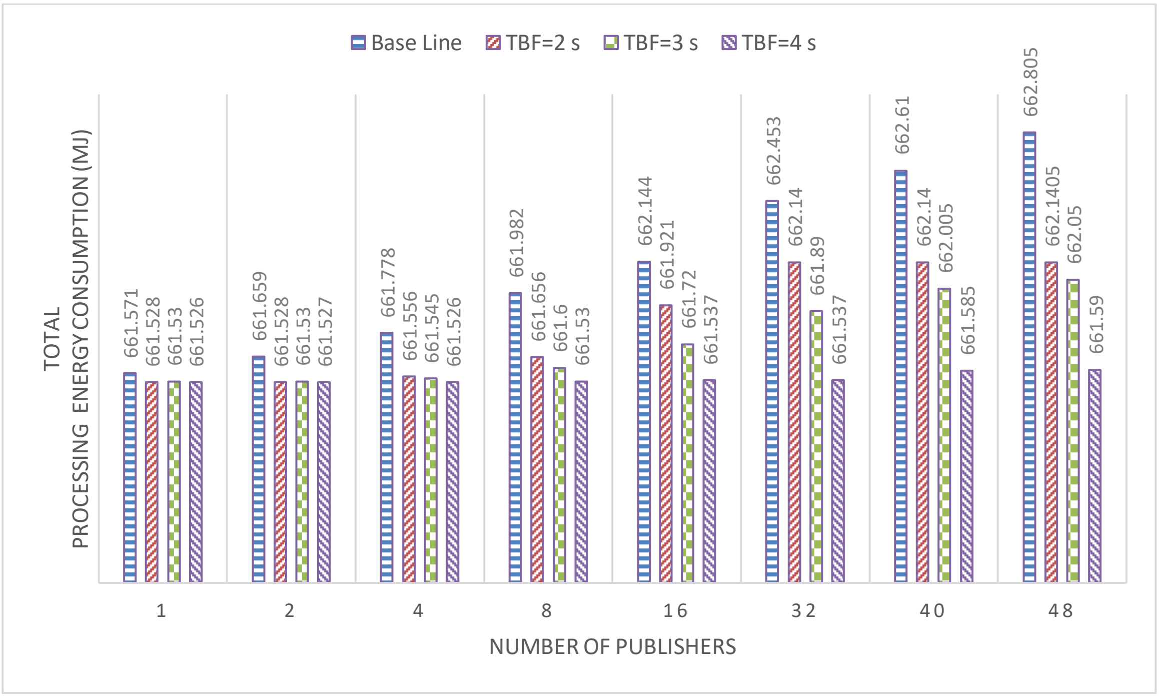


Figure 5.14: Total consumed processing energy for different TBF minimum separation (2 s, 3 s, and 4 s) while changing the number of publishers in data rate of 2 p/s

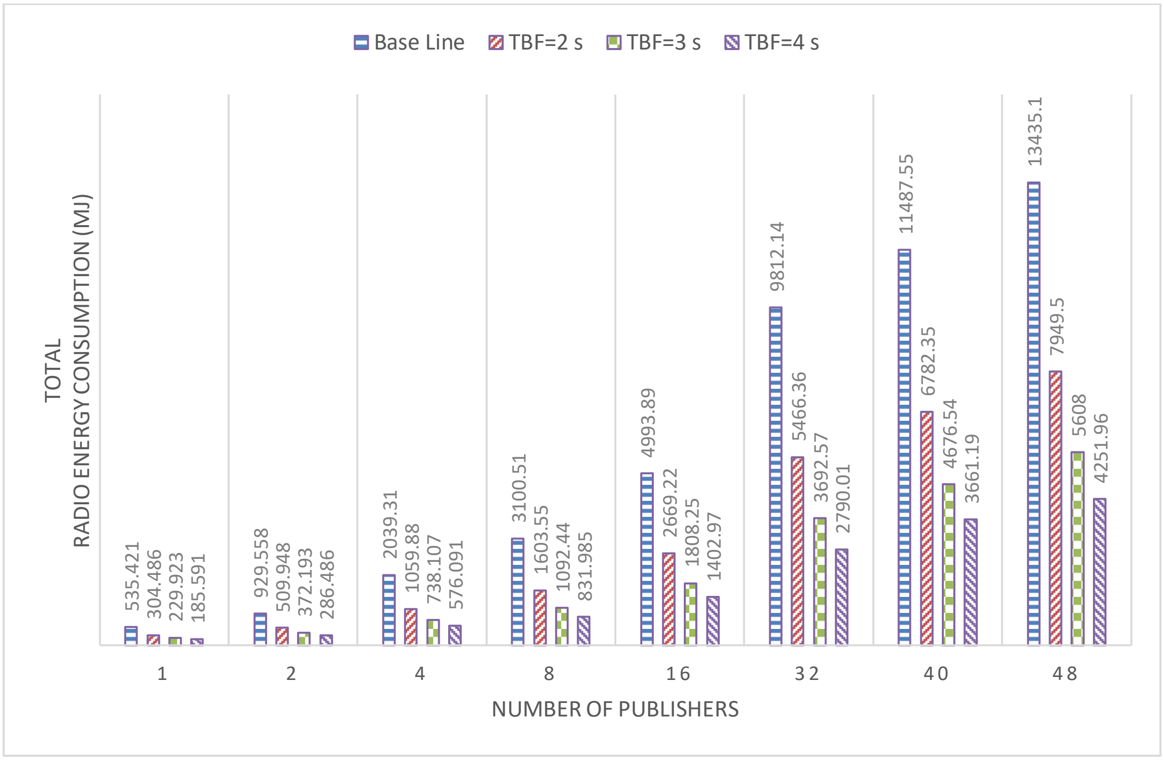


Figure 5.15: Total consumed radio energy for different TBF minimum separation (2 s, 3 s, and 4 s) while changing the number of publishers in data rate of 2 p/s.

5.2.3 Deadline QoS Policy results

In this section the Deadline QoS will be tested and the results will be shown, since Deadline QoS has two type which is offered Deadline, dedicated to the publisher, and Requested Deadline dedicated to the subscriber, the implementation require that both of these Deadline QoS's should work together for testing. We have mentioned before that offered deadline is a contract that application should meet and it defines the minimum time the application should prepare the data within for publishing. In our test scenario we have different publishers in each case, and this require to define different offered deadlines for each publisher. Table 5.1 show how offered deadline are assigned to each publisher in our test simulation.

Table 5.2: offered deadline assignment for each publishers in different number of publishers.

Number of publishers	Offered deadline (seconds)
1	1
2	1, 2
4	1, 2, 3, 4
8	1, 2, 3, 4, 1, 2, 3, 4
16	1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4
32	1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4
40	1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4
48	1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4

For requested Deadline the value is changed from (1, 2, and 3 seconds) and the results we got for different metrics are as follow:

Packet Delivery Ratio (PDR) Percentage.

From Figures 5.16 and 5.17 which represent the results of implementing Deadline QoS compared to the base Line results in different data rate (1 and 2 p/s). Since the requested Deadline is changed, this parameter defines weather the connections between the subscriber and different publishers can be established or not. We notice that in both figures when request deadline increases the PDR is decreased since that increasing request deadline will establish more connection and more data will be published to deliver to the subscriber and this will increase packet dropping and collisions and PDR will be decreased.

On the other hand decreasing the requested deadline will decrease the number of publishers that connects to the subscriber and this will lead to increase the PDR percentage. However we notice also increasing the number of publishers will increase the PDR in general to specific point where in full load, the increment will stop and more packet will not be delivered to the subscriber.

Deadline QoS implementation test shows improvement in system performance. For example, when number of publishers are 32 and request deadline = 1 s, the improvement is about 32.8% for 1 p/s data rate, and 40% for 2 p/s data rate. When request deadline = 2 s, the improvement is about 20.7% for 1 p/s data rate, and 20 % for 2 p/s data rate.

Deadline QoS improved the system performance, however decreasing the value of request Deadline will decrease amount of data delivered to the subscriber and less connection will be established and increasing the offered deadline will also decrease the amount of data to be published to that subscriber.

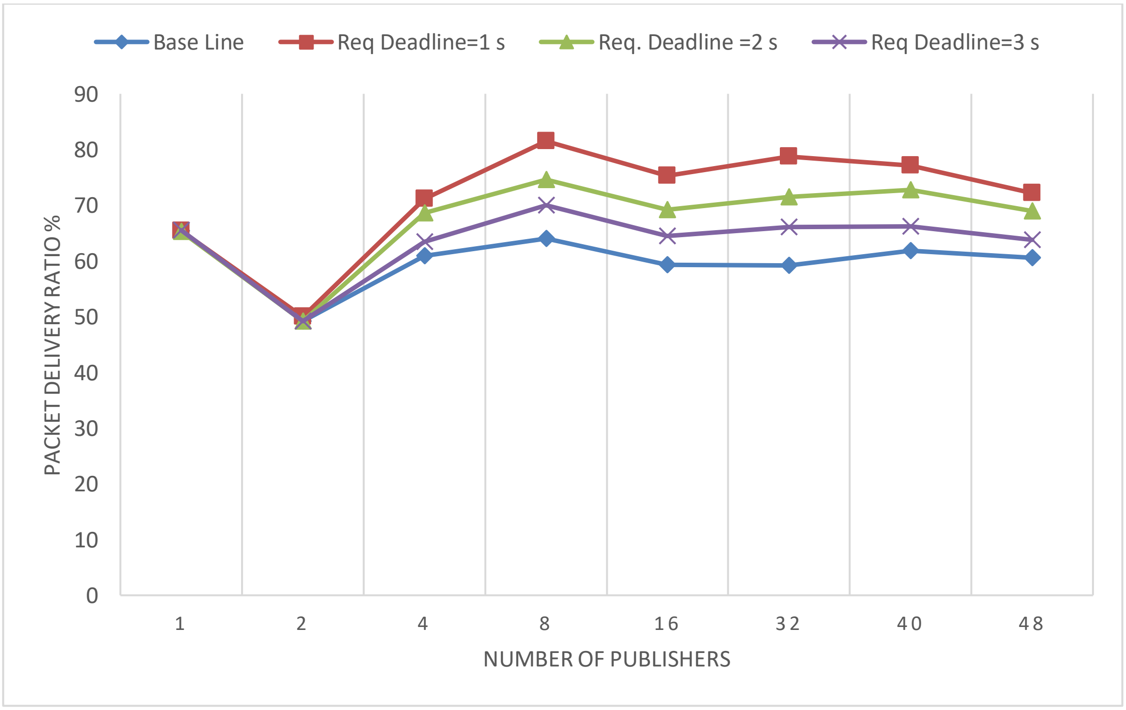


Figure 5.16: PDR behavior for different request deadline (1 s, 2 s, and 3 s)while changing the number of publishers in data rate of 1 p/s.

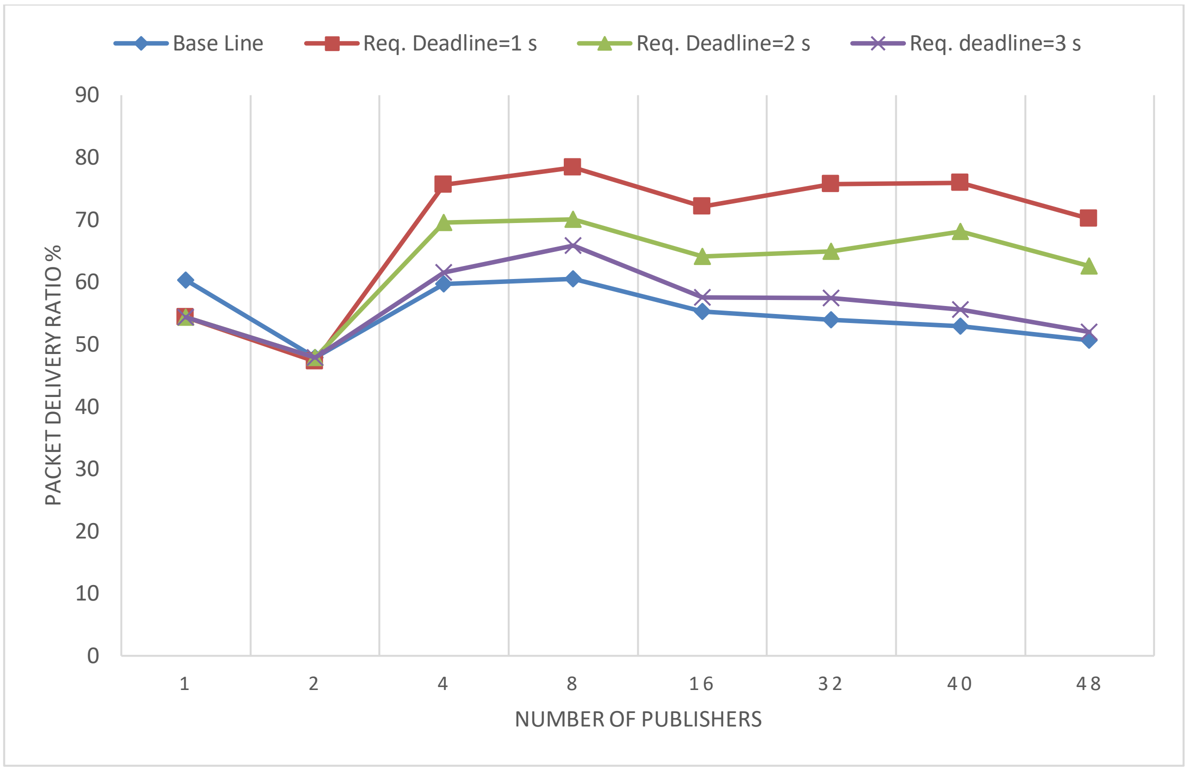


Figure 5.17: PDR behavior for different request deadline (1 s, 2 s, and 3 s)while changing the number of publishers in data rate of 2 p/s.

Latency:

Figures 5.18 and 5.19 illustrate the latency behavior for the system after implementing Deadline QoS for different data rate (1, 2 p/s). As we notice the latency of system is improved when implementing Deadline QoS whereas when the value of Deadline is low the improvement will increase, since less publishers will connect to the subscriber there will be less data traffic in the network. On other hand while increasing requested deadline, the effect was decreased and the improvement on the system was also decreased.

The results are also showing that the delay decreases as we increase number of publishers, however in case of more publishers than 32 the delay stops decreasing and starts increasing. When request Deadline = 1 s the delay improvement was maximum compared to others. However the improvement starts decreasing as we increase number of publishers and the network is fully loaded. For example, when number of publishers are 32 and request Deadline =1 s, the improvement in the latency is about 11.4 % for 1 p/s data rate, and 8.4 % for 2 p/s. when request deadline = 2 s, the improvement is 6.8 % for 1 p/s and 6 % for 2 p/s.

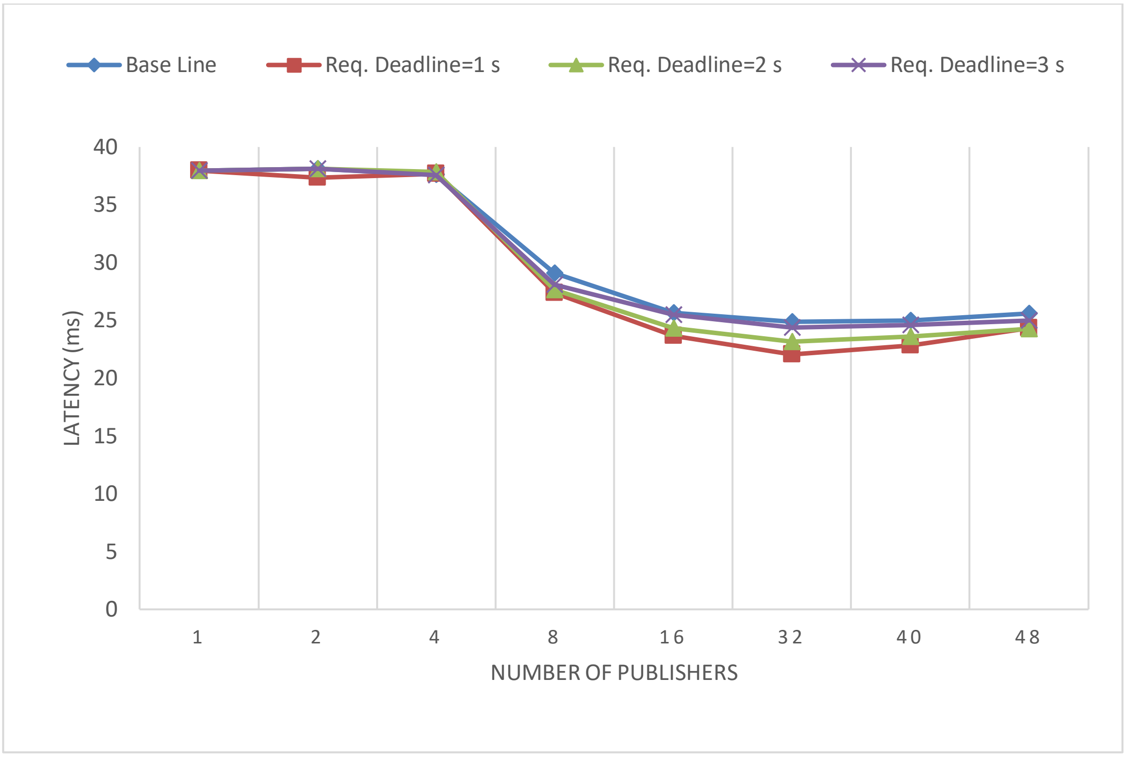


Figure 5.18: latency behavior for different request deadline (1 s, 2 s, and 3 s)while changing the number of publishers in data rate of 1 p/s.

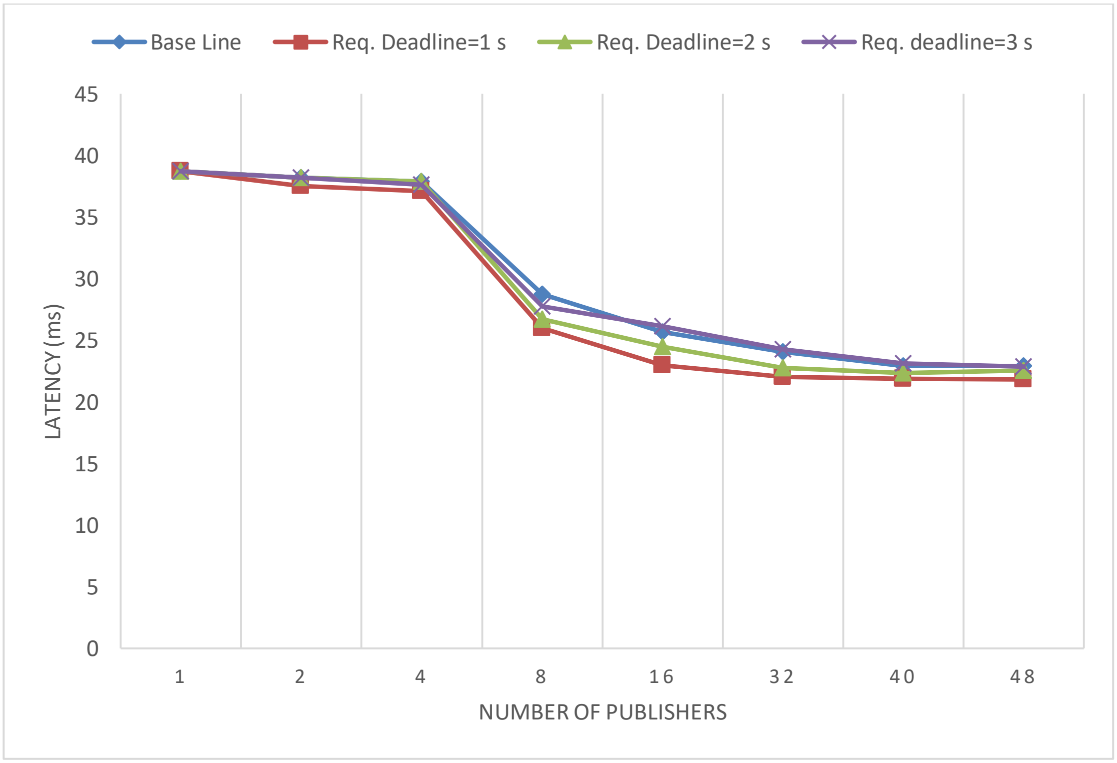


Figure 5.19: latency behavior for different request deadline (1 s, 2 s, and 3 s)while changing the number of publishers in data rate of 2 p/s.

Total Processing and Radio Energy Consumption:

Figure 5.20 and 5.21 show the results after applying Deadline QoS over energy consumption in the system and they compare it to the base line results for both energy type processing and radio in case of 1 p/s data rate. Also Figures 5.22 and 5.23 show the effect of applying TBF QoS in both types of energy when data rate is 2 p/s.

From Figures 5.20 and 5.22 we notice that the total processing energy consumed by all nodes in the network are decreased in case of 1, 2 and 3 seconds of request deadline and this returns to the effect of decreasing the packet forwarding mechanism since the publishing from all publishers is decreased. Added to that, the improvement in the total consumed processing energy was significant when increasing the number of publishers. For example, when number of publishers are 48, when request deadline = 1 seconds the total processing energy consumed by all nodes decreased by 0.04 % in 1 p/s and 0.13 % in 2 p/s. when request deadline = 2 seconds the total processing energy consumed by all nodes decreased by 0.02 % in 1 p/s and 0.06 % in 2 p/s.

In Figures 5.21 and 5.23 the results clarify that Deadline QoS decreases the total radio energy consumption significantly. The reason is that, applying Deadline QoS decreases the amount of traffic in the network and this will lead to less transmission in wireless medium that affect directly the radio energy for each node. The improvement is maximum when the value of request deadline is 1 second. For example, when number of publishers are 48 the total energy consumed by radio is decreased by 70% , when request deadline = 1 seconds and data rate =1 p/s and 67.5 % for the same request deadline and 2 p/s data rate. For the same number of publishers when request deadline =2 seconds and data rate is 1 p/s, the total energy consumed by all node radio is decreased by 48 % and 44.5 % when data rate is 2 p/s

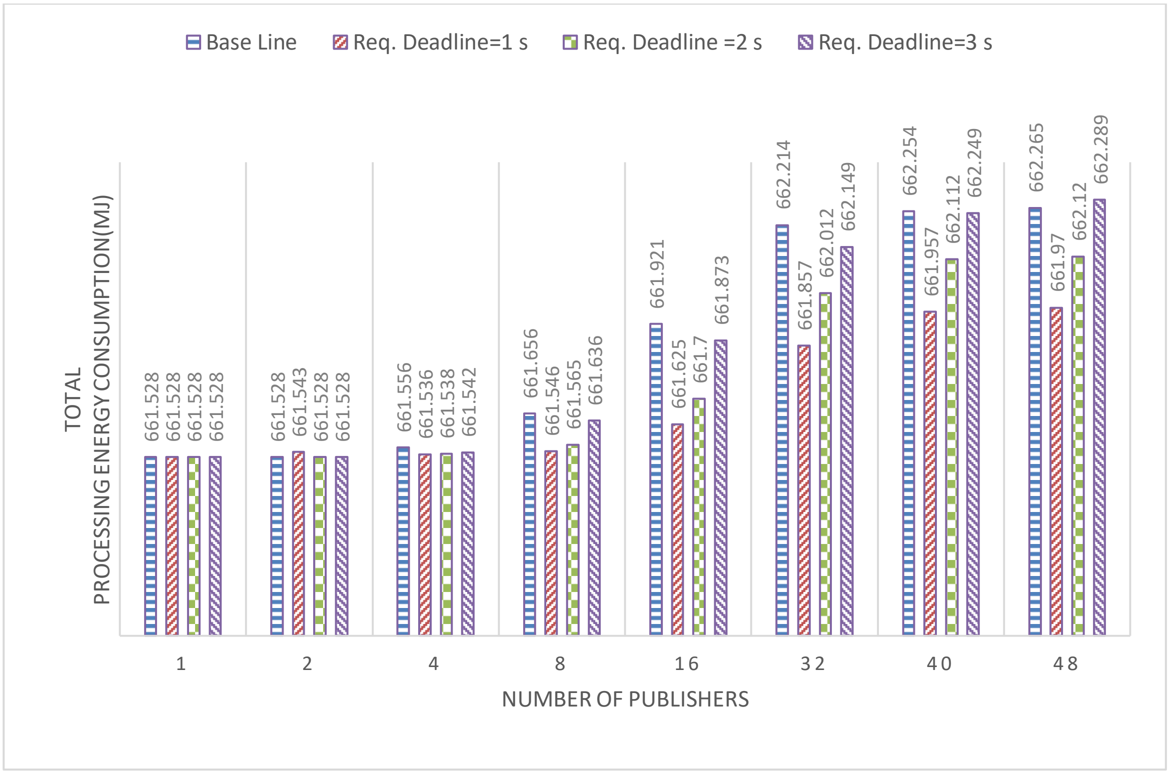


Figure 5.20: Total consumed energy behavior in processing for different requested deadline (1 s, 2 s, and 3 s) while changing the number of publishers in data rate of 1 p/s.

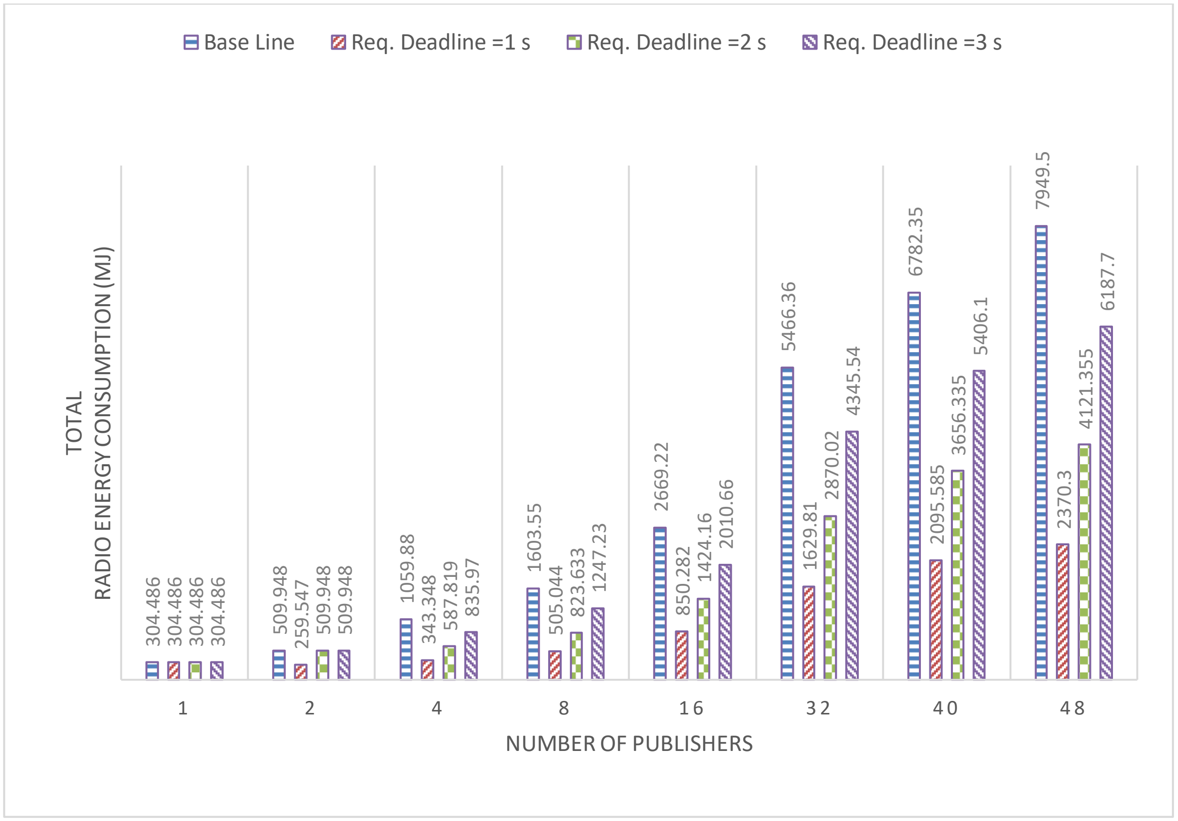


Figure 5.21: Total consumed energy behavior in radio for different requested deadline (1 s, 2 s, and 3 s) while changing the number of publishers in data rate of 1 p/s

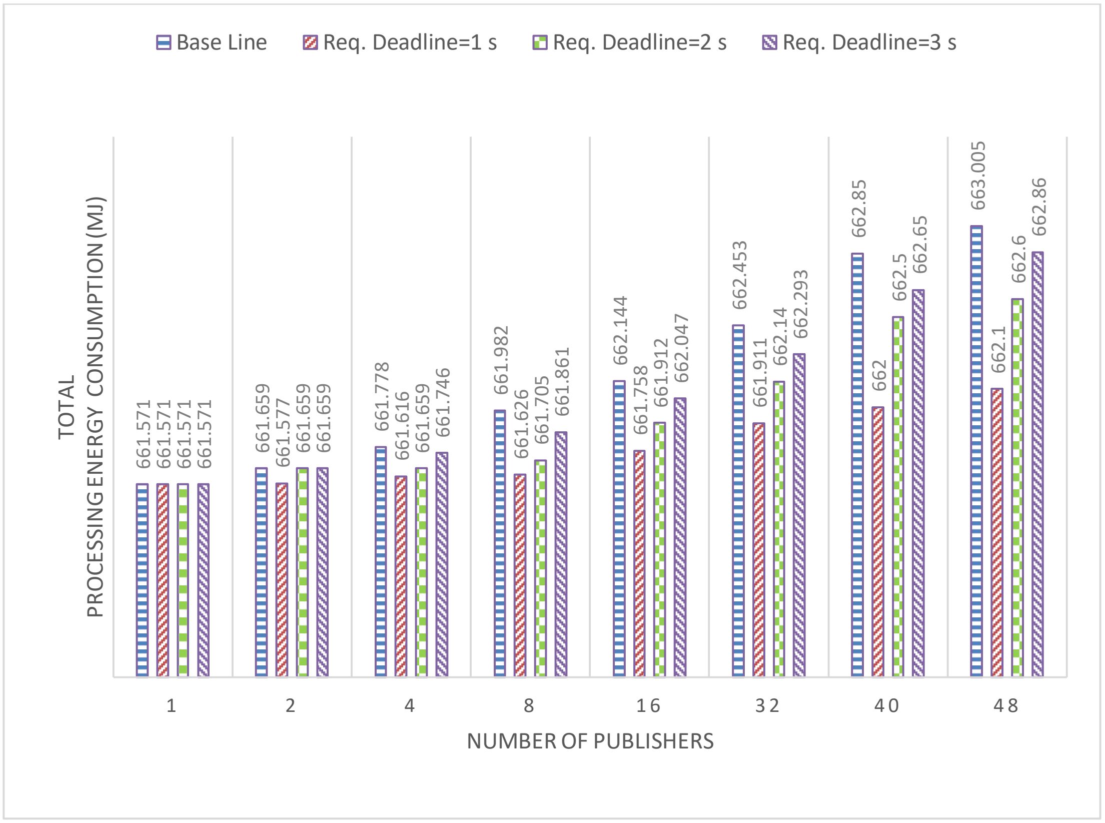


Figure 5.22: Total consumed energy behavior in processing for different requested deadline (1 s, 2 s, and 3 s) while changing the number of publishers in data rate of 2 p/s.

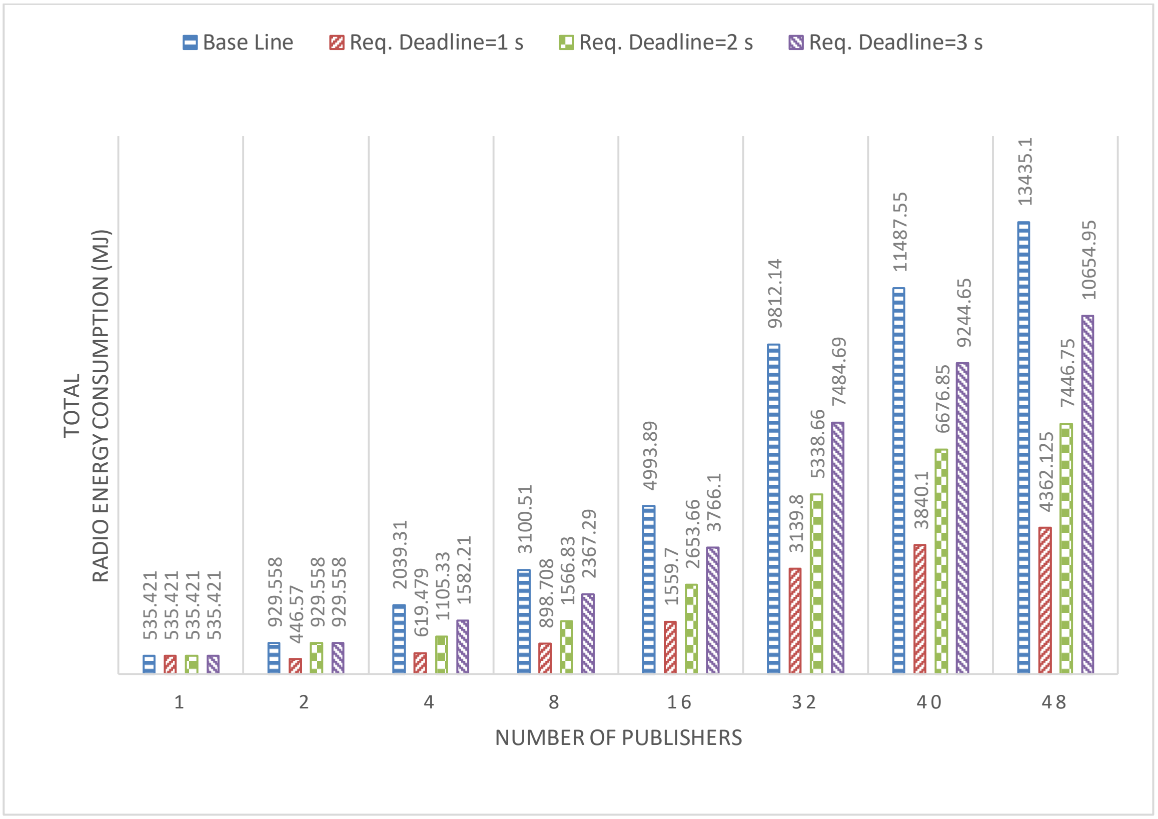


Figure 5.23: Total consumed energy behavior in radio for different requested deadline (1 s, 2 s, and 3 s) while changing the number of publishers in data rate of 2 p/s.

5.2.4 TBF and Deadline QoS's together results

In this section we will discuss the results when applying both of TBF and Deadline QoS over the system. We fixed the offered deadline parameter as in table 5.1. The requested deadline value in each figure simulation was fixed on a value of (3, 4, and 5 seconds) and for each figure we changed TBF minimum separations from (2 to 3 seconds).

We evaluated the system behavior for PDR and latency metrics and the results are as follow:

Packet Delivery Ratio (PDR) percentage

Figures 5.24, 5.25 and 5.26 show the system PDR percentage changes for different requested deadlines and TBF values in 1 p/s data rate. From the figures we notice that applying both QoS's was improving the system performance. However, it decreases the mount of published data. From figure 5.24 we notice the effect of consistency between TBF and Deadline whereas communication does not occur between publishers and the subscriber since TBF = 2 s and the requested deadline = 3. In figures 5.25 and 5.26 the consistency issue is passed successfully for all values of TBF QoS. Although the results in all figures show that the performance will increase in case of decreasing the requested deadline and this improvements is more than the one related to changing TBF QoS minimum separation time values. It also decreases the amount of published data to the base station.

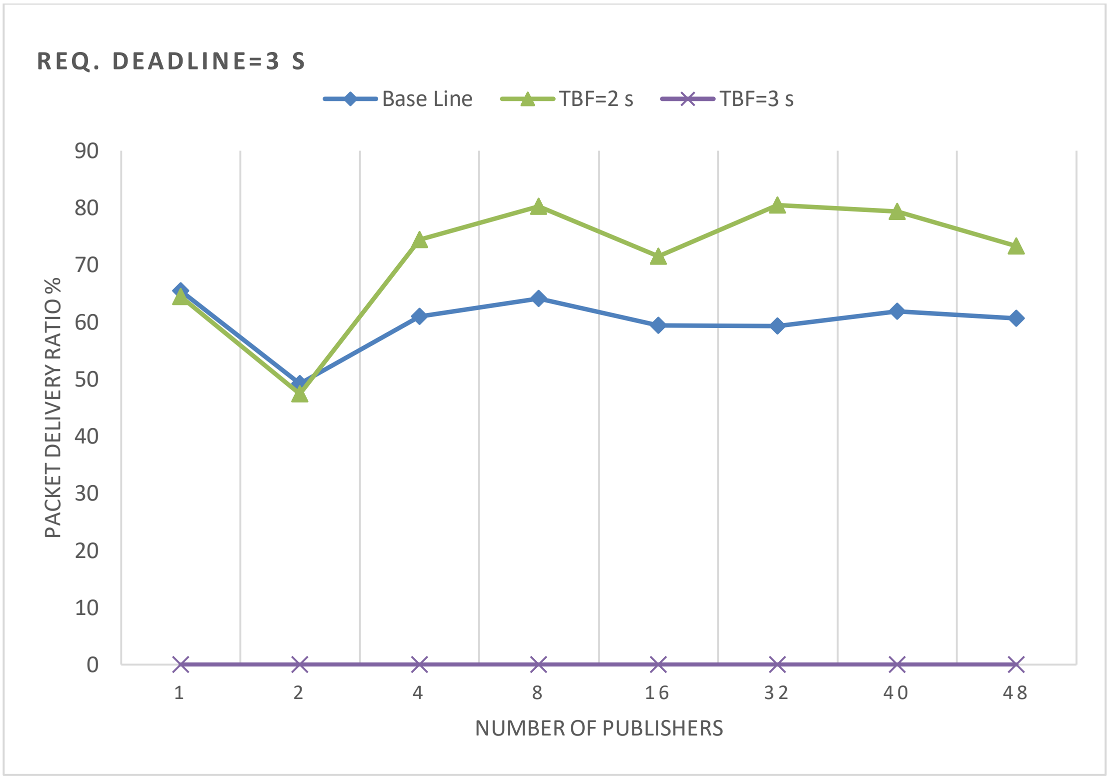


Figure 5.24: PDR behavior when changing TBF minimum separation time (2 s, and 3 s) while changing the number of publishers, and the requested deadline = 3 s and 1 p/s data rate.

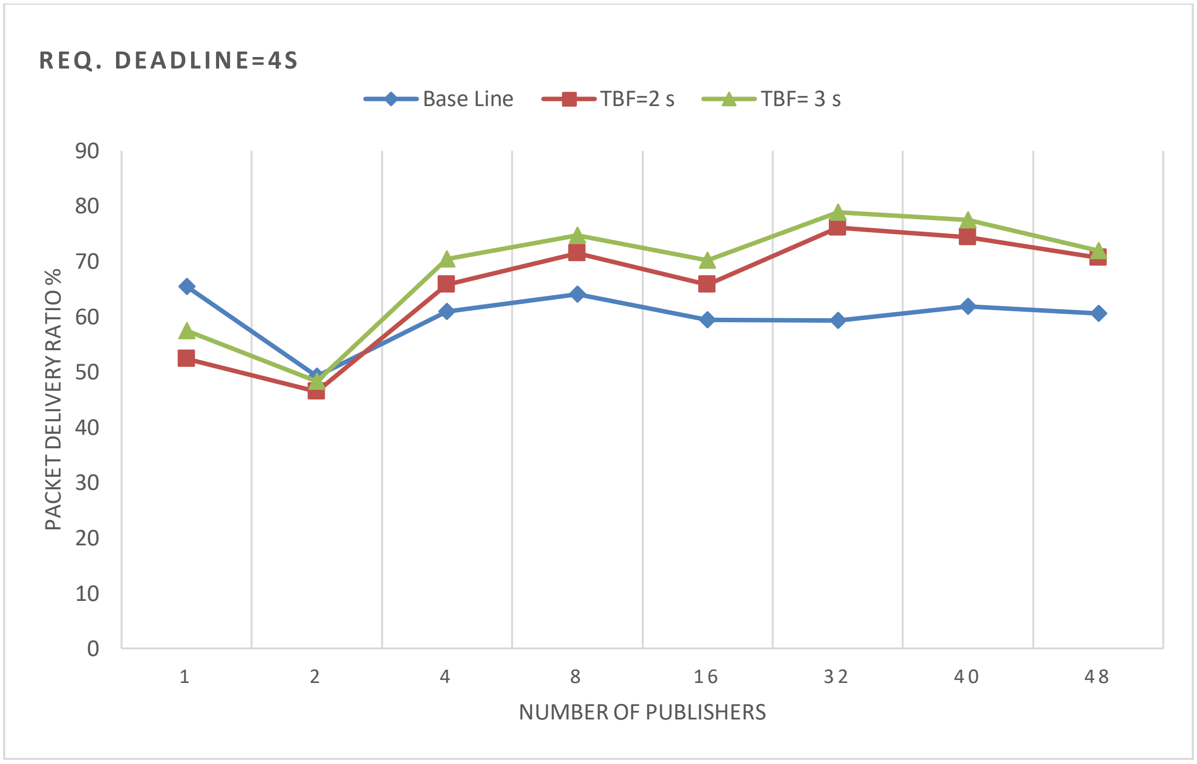


Figure 5.25: PDR behavior when changing TBF minimum separation time (2 s, and 3 s) while changing the number of publishers, and the requested deadline = 4 s and 1 p/s data rate.

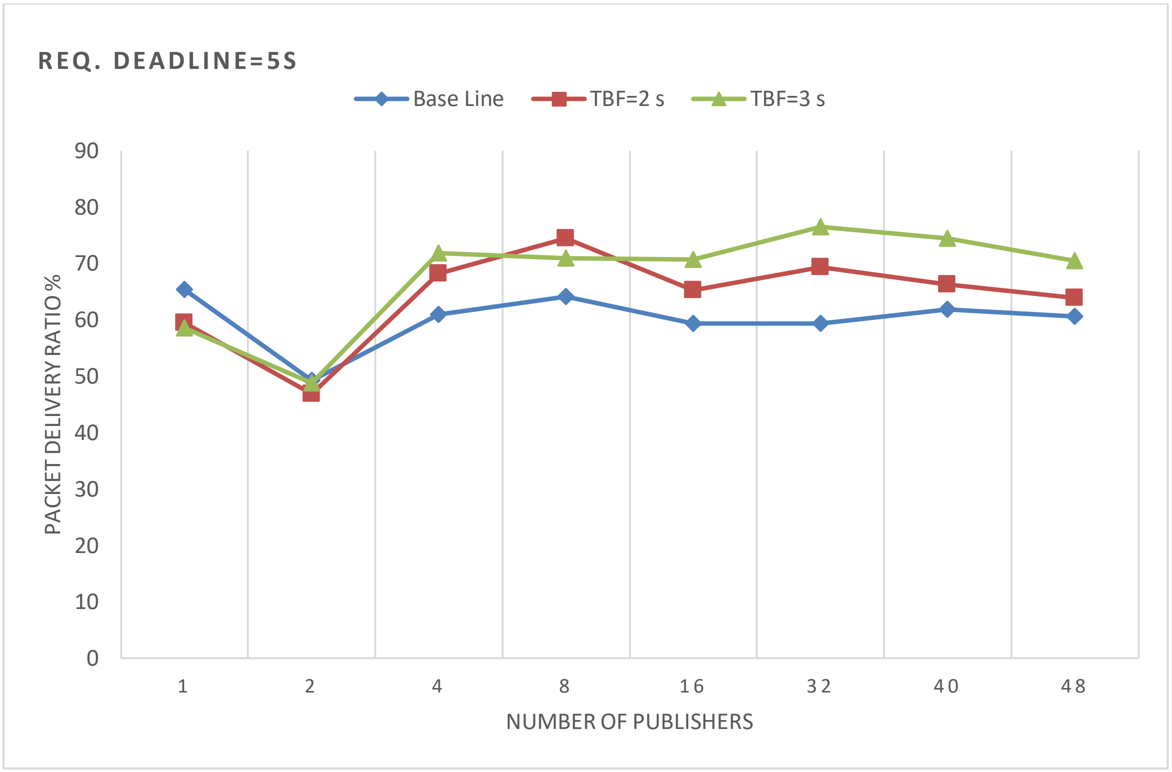


Figure 5.26: PDR behavior when changing TBF minimum separation time (2 s, and 3 s)while changing the number of publishers, and the requested deadline = 5 s and 1 p/s data rate.

Latency:

Figures 5.27, 5.28 and 5.29 show the system latency changes for different requested deadlines and TBF values in 1 p/s data rate. From the figures we notice that applying both QoSs is improving the system performance in case of 1 to 32 publishers however the latency increased for more than 32 publishers. In Figure 5.27 due to consistency issue between the QoSs, the communication doesn't occur, in case of TBF =2s. However, the communication occurred for all values of TBF in other figures.

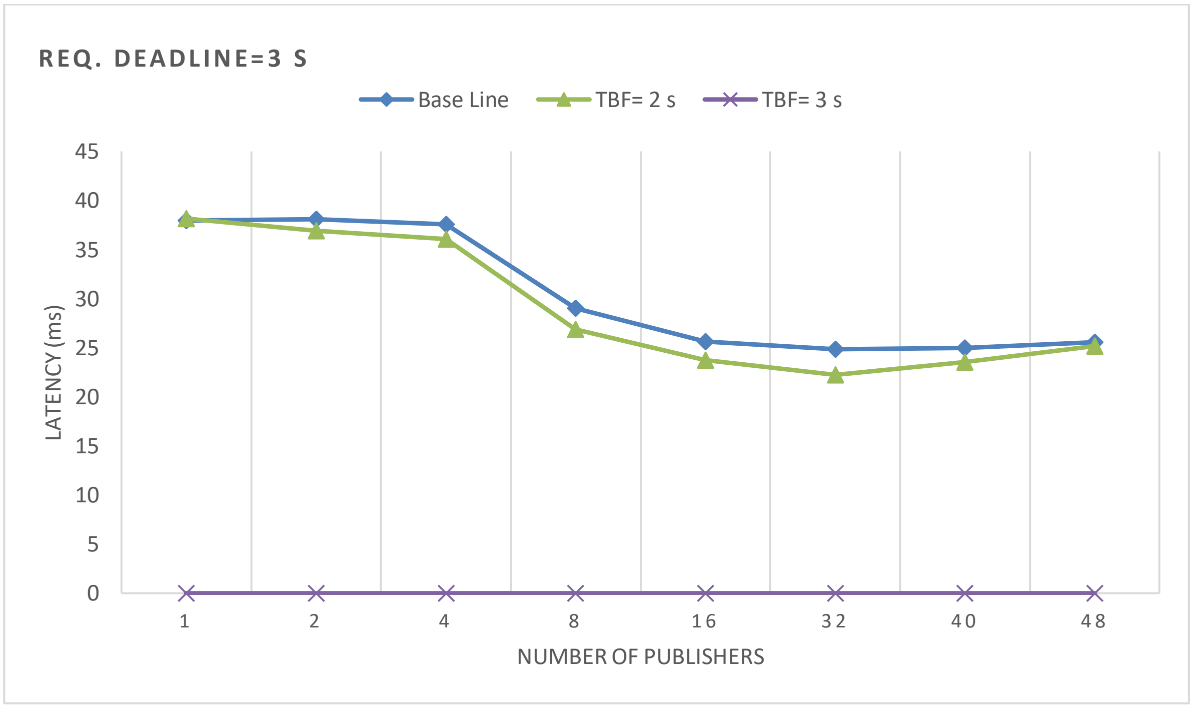


Figure 5.27: latency behavior when changing TBF minimum separation time (2 s, and 3 s) while changing the number of publishers, and the requested deadline = 3 s and 1 p/s data rate.

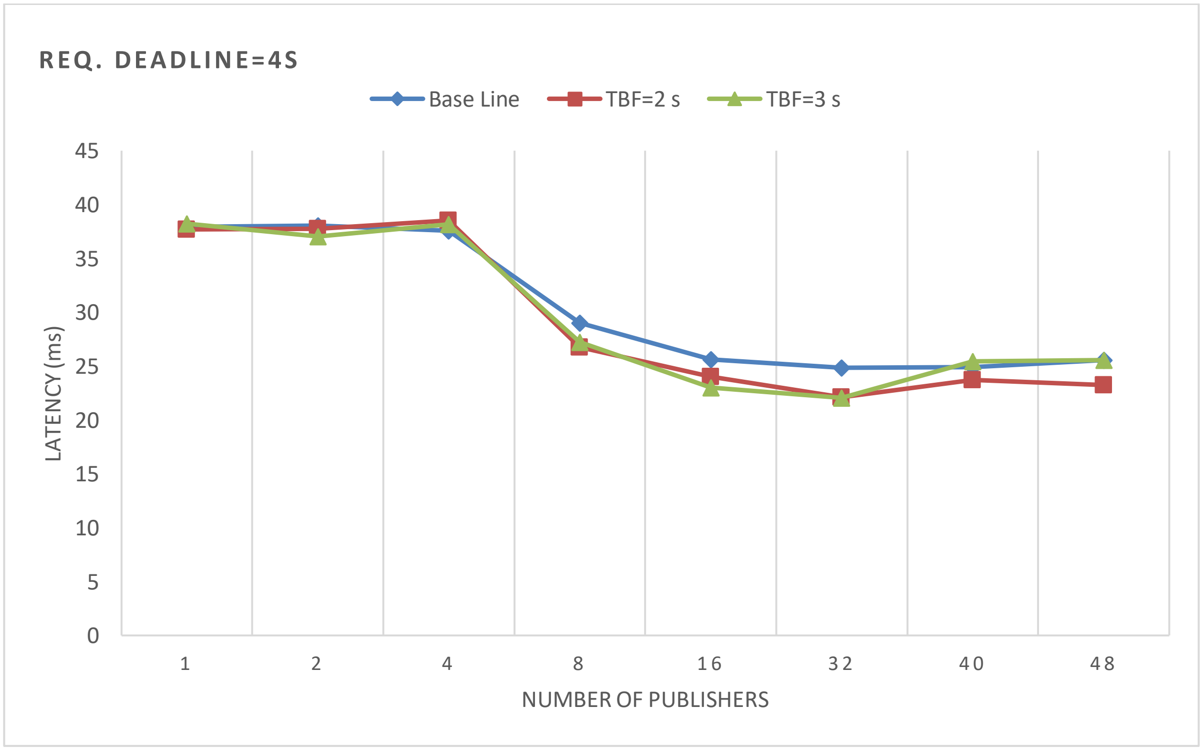


Figure 5.28: latency behavior when changing TBF minimum separation time (2 s, and 3 s) while changing the number of publishers, and the requested deadline = 4 s and 1 p/s data rate.

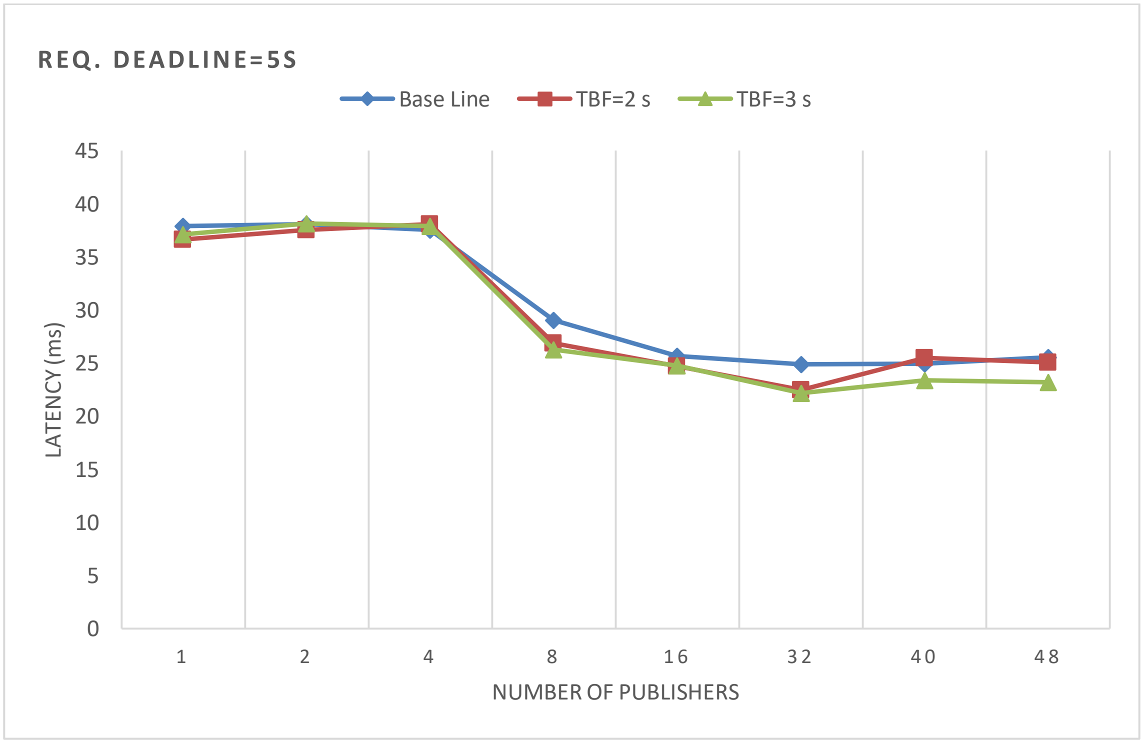


Figure 5.29: latency behavior when changing TBF minimum separation time (2 s, and 3 s) while changing the number of publishers, and the requested deadline = 5 s and 1 p/s data rate.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

TinyDDS middleware is emerging as state of the art middleware for WSANs since it is based on the OMG DDS standard. The improved version of TinyDDS, which is Broker-less TinyDDS, has been used is this work since it does not depend on a broker to deliver the data between publishers and subscribers. DDS real-time QoSs such as Time Based Filter and Deadline are supported but not implemented in BLTDDS. In this work, the implementation of Time Based Filter and Deadline has been done and tested and evaluated. Time Based Filter QoS provides minimum separation time between data samples to cope with the subscriber Data reader's limited resources, and it decreases the traffic in the network and increases the available bandwidth and delivery ratio. In addition, it improves the packet latency and support the scalability in the network. In addition, the Deadline QoS is implemented to assure data availability within specific time in the publisher side and determines the maximum inter arrival time for data samples in the subscriber side. Thus it is an agreement for communication to occur between publishers and the subscriber. Deadline QoS also improves the system performance and latency. However, both QoSs decrease the amount of publish data in the system, so that it should be assigned in a way which is more related to the application needs to avoid the consistency issue. Packet Delivery Ratio, Latency and Energy Consumption are the metrics used to test and evaluate the implementation.

6.2 Future work

The future work improvements will look into the following aspects:

1- Since Broker-Less TinyDDS middleware is proposed as the perfect middleware for WSANs it still lacks the implementation of other DDS QoSs, such as Ownership, transport priority, and others which will guarantee more quality of service in the network and improve its performance.

2- To go ahead and do an experimental test for these middleware over sensor nodes and to compare the result to the simulation ones.

3- To implement these QoSs for default TinyDDS and to compare the results with simulation results of BLTDDS.

4- To increase the subscriber nodes in the network and evaluate the performance before and after implement these and new QoSs.

5- To do comparisons between BLTDDS with real time QoS and other middleware technologies that are also designed for WSANs to compare the costs and performance.

6- To use different network topologies in the simulation and other simulation tools to validate the results and feedback and to also add different heterogeneous network and check the middleware performance.

7- To study more network performance measures such as memory footprint and jitter.

8- TinyDDS is an open source middleware that needs more improvements in term of routing protocols and packet forwarding mechanisms [20].

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] Akyildiz, I. F., & Kasimoglu, I. H. (2004). **Wireless sensor and actor networks: Research challenges. Ad Hoc Networks**, 2(4), 351–367.

[3] Gupta, K., Sikka, V., "**Design Issues and Challenges in Wireless Sensor Networks**" *International Journal of Computer Applications*, vol. 112, no. 4, February 2015.

[4] Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.-M., "**The many faces of pub/sub**". *ACM Computing Surveys* v. 35 (2), pp.114–131, 2003

[5] Hamdan D., Aktouf O., Parissis I., Hassan B.E., Hijazi A. and Moslem B., "**A Self-Monitoring, Adaptive and Resource Efficient Approach for Improving QoS in Wireless Sensor Networks,**" *in 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC).*

[6] Karimi. H., Kargahi. M. and Yazdani. N., "**On the Handling Node Failure: Energy-Efficient Job Allocation Algorithm for Real-Time Sensor Networks**," *in 4th International Conference on Embedded and Multimedia Computing,* 2009. EM-Com, 2009.

[7] T. S. Prakash, K. Raja, K. Venugopal, S. Lyengar and L. Patnaik, "**Fault Tolerant QoS Adaptive Clustering for Wireless Sensor Networks,**" *in Proceedings of Ninth International Conference on Wireless Communication and Sensor Networks*, India 2014

[8] Hoffert J., Schmidt D., and Gokhale A., "DQML: **a modeling language for configuring distributed pub/sub quality of service policies**," *The 10th International Symposium on Distributed Objects, Middleware, and Applications, DOA '08*, Monterrey, Mexico, November 2008.

[9] Hoffert J., Mack D., and Schmidt D., "**Integrating machine learning techniques to adapt protocols for QoS-enabled distributed real-time and embedded pub/sub middleware**," *International Journal of Network Protocols and Algorithms (NPA): Special Issue on Data Dissemination for Large-scale Complex Critical Infrastructures*, vol. 2, no. 3, 2010.

[10] Hoffert J., and Schmidt D., "**Adapting distributed real-time and embedded pub/sub middleware for cloud-computing environments**," *Proc. ACM/IFIP/USENIX 11th International Middleware Conference*, Bangalore, India, November 2010.

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

[12] F. Xia, "**QoS Challenges and Opportunities in Wireless Sensor/Actuator Networks**," *Sensors*, vol. 8, no. 2, pp. 1099-1110, 2008.

[13] 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.

[14] 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.

[15] 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.

[16] Henricksen, K., & Robinson, R. (2006) **A survey of middleware for sensor networks: State-of-the-art and future directions**. *In Proceedings of the international workshop on Middleware for sensor networks.*

[17] 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.

[18] Object Management Group, "**Data Distribution Services (DDS),**" V1.4, OMG specifications 2015. [Online]. Available: http://www.omg.org/spec/DDS/ [Accessed October 2016].

[19] **RTI connext DDS software for real-time systems**, 2013. [Online]. Available: http://www.rti.com/products/dds/index.html [Accessed October 2016].

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

[21] González, A.; Mata, W.; Villaseñor, L.; Aquino, R.; Simó Ten, JE.; Chávez, M.; Crespo Lorente, A., " **uDDS: A Middleware for Real-time Wireless Embedded Systems**". *Journal of Intelligent and Robotic Systems*, vol. 64, no. 3-4, pp. 489-503, 2011

[22] Sheltami, T. R., Al-Roubaiey, A. A., Mahmoud, A. S., "**A survey on developing pub/sub middleware over wireless sensor/actuator**" *Springer Science plus Business Media New York, Wireless Netw*, 2015.

[23] Sheltami, T. R., Al-Roubaiey, A. A., Mahmoud, A. S., Shakshuki, E., "**A Pub/sub Middleware Cost in Wireless Sensor Networks, A review and case study**", *Proceeding of the IEEE 28th Canadian Conference on Electrical and Computer Engineering*, May 3-6, 2015.

[24] Kumar, V., Gupta, A. K., "**Measuring Parameters of Quality of Service in Wireless Sensor Networks**", *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 3, issue 11, November 2014

[25] Pérez, H., Gutiérrez, J., J., "**Modeling the QoS parameters of DDS for event-driven real-time applications**", *The Journal of Systems and Software*, vol. 104, pp. 126-140, 2015.

[26] Kaur, S., Mir, R., N., "**Quality of Service in WSN-A Review**", *International Journal of Computer Applications*, vol. 113, no. 18, March 2015

[27] Bamatraf, A., Bin Abd latiff, M., S., Coulibaly, Y., Khasawneh, A., M., "**Review of quality of service in routing protocols for wireless sensor networks**", *Journal of Theoretical and Applied Information Technology*, vol. 74, no.3, April 2015.

[28] Schlesselman, J., M., PardoCastellote, G., and Farabaugh, B., "**OMG data-distribution service (DDS): architectural update**." *Military Communications Conference MILCOM IEEE. vol. 2*. 2004.

[29] Essers, M.S., Vanekera, T.H.J., "**Evaluating a prototype approach to validating a DDS-based system architecture for automated manufacturing environments**"*8th International Conference on Digital Enterprise Technology - DET*2014.

[30] **TOSSIM, "TinyOS Documentation Wiki,"** 2003. [Online]. Available: http://docs.tinyos.net/index.php/TOSSIM. [Accessed October 2016].

[31] 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

Vitae

Name	: Samer Khaled Yousef Rabah
Nationality	: Palestine
Date of Birth	: 9/18/1989
Email	: samer.rabah1@hotmail.com
Address	: Hebron-Palestine
Academic Background	: Complete M.Sc. in Computer Networks from Computer Engineering department at King Fahd University of Petroleum and Minerals(KFUPM) at Dec 2016 , earned the B.S. degree in Communication and Electronics Engineering from Palestine Polytechnic University Hebron-Palestine 2012. Research interests include wireless sensor networks, routing protocols, data distribution system, middlewares, heterogeneous networks, also multi input multi output antenna, orthogonal frequency division multiplexing, digital communications and signal processing.